P6066

Manuale generale

olivetti

GR Code 3946860 Y (0)

PREFAZIONE

La presente pubblicazione spiega come si usa il minicomputer Olivetti P6066, un sistema general-purpose programmabile in BASIC. Il manuale è diretto soprattutto ad ingegneri, matematici, scienziati ed a chiunque conosca il linguaggio BASIC, ma non sia un programmatore professionista, a coloro, cioè, che sono interessati alla programmazione intesa come modo di risolvere i problemi (elaborazione di dati veloce, accurata ed intelligente).

Questo manuale fornisce tutte le informazioni necessarie per trattare i file dati e file testo e creare, stampare, eseguire, verificare ed archiviare programmi sul sistema P6066. Sono descritti ed illustrati con esempi i comandi, le istruzioni, i programmi di utilità e i modi operativi. Una parte del manuale è dedicata ai messaggi inviati dal sistema ed ai messaggi di errore; per ognuno di questi ultimi è illustrata anche la causa dell'errore. In aggiunta, la pubblicazione contiene anche informazioni sull'operabilità del sistema, come, ad esempio, l'uso dell'unità floppy disk e della stampante termica.

Poichè il manuale ha lo scopo di insegnare al lettore come usare il sistema e non di spiegare come esso è strutturato, si è evitato, per quanto possibile, di addentrarsi in specifiche tecniche.

Al lettore che non ha mai programmato od usato il linguaggio BASIC si consiglia di leggere prima di questo manuale la pubblicazione P6066 Introduzione. L'Introduzione fornisce una panoramica sia del linguaggio BASIC che del P6066.

© 1980, by Olivetti

Riferimenti:

P6066 - Introduzione - GD Code 3978520 Z

P6066 - I/O con periferiche esterne
Manuale del programmatore
GP Code 3978330 N

P6066 - Prestazioni grafiche
GP Code 3978350 Q

P6066 - Gestione dei file testo
Guida dell'utente - GD Code 3977870 Y

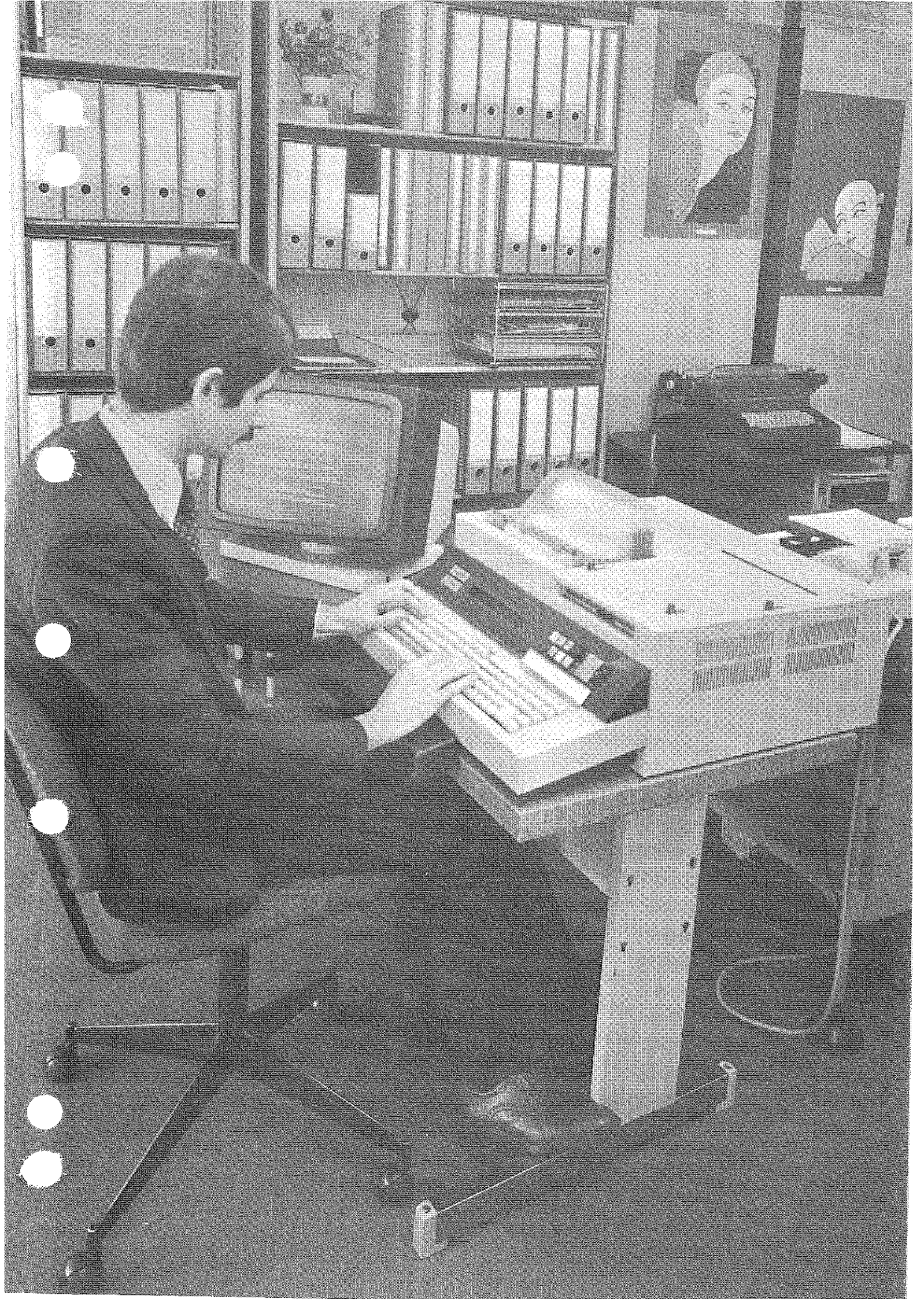
Distribuzione : Generale

Prima edizione : Gennaio 1980

PUBBLICAZIONE EMESSA DA:

Ing. C. Olivetti & C., S.p.A.
Direzione Marketing Centrale
Servizio Documentazione
77, Via Jervis - 10015 IVREA (Italy)

3946860 Y



INDICE

INTRODUZIONE xiii

1. IL SISTEMA P6066 1-1

Configurazioni minime di sistema 1-1

Unità centrale di elaborazione 1-3
Memoria principale 1-3
Memoria a sola lettura 1-4
Governi di input/output 1-4
Tastiera 1-4
La console 1-13
Il display 1-15
Unità floppy disk 1-16
Segnalatore acustico 1-17
Interruttore 1-18
Unità a disco 1-18

Configurazioni estese 1-24

2. COME USARE IL SISTEMA 2-1

Preparazione dei supporti 2-1

Cambio del rullo di carta 2-1
Inserimento del floppy disk 2-4

Accensione 2-6

Inizializzazione 2-6

Introduzione da tastiera 2-7

Correzione delle introduzioni da tastiera 2-9

Introduzione della data 2-10

Working File 2-10

Opzioni del sistema operativo 2-10

Stati del sistema 2-12

Stato comandi 2-12
Stato di esecuzione programma 2-13
Stato di debugging 2-15
Stato di esecuzione calcoli immediati 2-15

Creazione ed editing di un programma BASIC 2-15

Struttura di un programma BASIC 2-15

Introduzione ed editing di un programma 2-16

Creazione ed editing di un file testo 2-21

Struttura di un file testo 2-21
Introduzione ed editing di un testo 2-22

Spegnimento 2-23

3. COMANDI DI SISTEMA 3-1

I dischi 3-1

I floppy disk 3-2

Nomi per le unità a disco e floppy disk 3-2

Organizzazione delle informazioni su disco o floppy disk 3-3

Librerie 3-3

Nomi e parole d'ordine 3-4
Sottolibrerie 3-5
File 3-6

Introduzione di un comando 3-7

Notazioni 3-8

Elenco e funzione dei comandi di sistema 3-9

AUTO# 3-13
CATALOG 3-17
COMPILE 3-23
CONFIGURE 3-25
CREATE 3-29
DATE 3-33
DCHANGE 3-35
DECOMPILE 3-37
DELETE LINE 3-39
ENVIRONMENT 3-41
EXEC 3-43
FETCH 3-45
LBCLOSE 3-47
LBOPEN 3-49
LBRESTORE 3-51
LBSTORE 3-53
LDKEYS 3-55
LINK 3-57
LIST 3-61
LVTOC 3-65
MERGE 3-69
MODIFY 3-75
NEW 3-79
OLD 3-81
OPTIONS 3-83
PREPARE 3-87

PROCEDURE CATALOGATE 3-91

Generalità 3-91

Salti condizionati 3-94

Procedura avviata alla inizializzazione del sistema 3-94

Testi di una procedura: comando TEST 3-94

Esempio 3-95

PROCEDURE 3-97
PURGE 3-103
REPLACE 3-105
RESEQUENCE 3-107
RUN 3-111
SAVE 3-115
SECURE 3-119
SHIFT 3-123
SPACE 3-125
START 3-127
STKEYS 3-129
STOP 3-131
TEXT 3-133
TRANSCODE 3-135
TRUNCATE 3-139
VALIDATE 3-141

4. BASIC: DATI, VARIABILI, ESPRESSIONE E FILE DATI 4-1

I caratteri del linguaggio BASIC 4-1

Caratteri alfabetici 4-1
Caratteri numerici 4-1
Caratteri speciali 4-1
Gli spazi 4-2

Dati numerici 4-3

Grandezza di un numero 4-3
Precisione 4-3
Formato dei dati numerici 4-5
Costanti numeriche 4-6
Variabili semplici numeriche 4-7

Dati non numerici 4-8

Costanti stringa 4-8
Variabili semplici stringa 4-9

Variabili multiple 4-10

Dichiarazioni delle variabili multiple 4-11

Ridimensionamento delle variabili multiple 4-13

Variabili multiple numeriche 4-14

Variabili multiple stringa 4-16

Nomi delle variabili 4-17

Area comune 4-18

Funzioni di sistema 4-18

Funzioni numeriche di sistema 4-18
Funzioni stringa di sistema 4-20
Funzione speciale di sistema 4-30

Espressioni 4-30

Espressioni numeriche 4-31
Operatori numerici 4-32
Espressioni stringa e operatori 4-34
Espressioni di confronto 4-35
Espressioni con variabili multiple numeriche 4-36

File dati 4-36

File dati interni 4-37
File dati esterni 4-38

5. LE ISTRUZIONI BASIC 5-1

Il programma BASIC e le istruzioni BASIC 5-1

Introduzione di linee BASIC 5-2

Notazioni 5-3

Elenco e funzione delle istruzioni BASIC 5-4

Descrizione delle istruzioni BASIC 5-9

APPEND: 5-11
ASSIGN 5-13
BASSIGN 5-17
BBUILD 5-19
BEEP 5-23
BPAD 5-25
BUILD 5-29
BUILD USING 5-33
CALL 5-37
CHAIN 5-39
COMMON 5-43
CONVERT 5-49
DATA 5-53
DCL 5-59
DEF 5-63
DEF/FNEND 5-69
DELAY 5-79
DEPAD 5-81
DIM 5-83
DISP 5-87
DISP USING 5-97
END 5-103
FILE: 5-105
FILES 5-109
FKEY 5-113
FNEND 5-119
FOR 5-121
GO SUB 5-129
GO TO 5-133
IF ... THEN 5-137
Istruzione IMMAGINE 5-143
INPUT 5-151
INTERRUPT ENABLE 5-157
LET 5-161
NEXT 5-165
ON ... GOSUB 5-167
ON ... GOTO 5-171
PAD 5-175
PRINT 5-179
PRINT USING 5-189
RANDOMIZE 5-193
READ 5-195
READ: 5-197
REMARK 5-205
RESTORE 5-207
RESTORE: 5-209
RETURN 5-213
RKB 5-215
SCRATCH: 5-217
SETW: 5-219

STOP 5-223
TRACE OFF 5-225
TRACE ON 5-227
WHERE: 5-231
WRITE: 5-235

Istruzioni BASIC per l'elaborazione di matrici 5-243

MAT ... = 5-247
MAT ... + 5-251
MAT ... - 5-255
MAT ... * (moltip scalare) 5-257
MAT ... * 5-259
MAT ... CON 5-263
MAT ... IDN 5-265
MAT ... INV 5-267
MAT ... TRN 5-271
MAT ... ZER 5-273
MAT INPUT 5-275
MAT PRINT 5-279
MAT PRINT USING 5-285
MAT READ 5-291
MAT READ: 5-295
MAT WRITE: 5-301

6. STATO CALCOLI IMMEDIATI 6-1

Introduzione ed esecuzione di espressioni nello stato calcoli immediati 6-2

Espressioni numeriche nello stato calcoli immediati 6-2

Costanti numeriche 6-2

Formato intero 6-3
Formato in virgola fissa 6-3
Formato in virgola mobile 6-3

Rappresentazione interna 6-3

Campo della rappresentazione interna 6-3

Variabili numeriche 6-5

La variabile Φ come totalizzatore 6-6

Operatori numerici 6-7

Funzioni 6-10

Funzioni numeriche di sistema 6-10
Funzioni definite dall'utente 6-11

Visualizzazione dei risultati 6-12

L'unità di misura degli angoli 6-13

Esempi di calcoli immediati 6-14

7. LO STATO DI DEBUGGING 7-1

Premessa 7-1

Come accedere allo stato di debugging 7-1

Strumenti dello stato di debugging 7-2

Comandi dello stato di debugging 7-3

Tasti di console 7-4

Prestazioni dello stato calcoli immediati 7-5

Analisi ed impiego delle variabili globali di un programma 7-6

Impiego delle funzioni definite nel programma 7-6

Un esempio di debugging di un programma 7-7

8. IMPIEGO DEI TASTI FUNZIONE 8-1

Assegnazione di una funzione ad un tasto funzione 8-1

Assegnazione di una funzione ai tasti funzione durante la esecuzione di un programma 8-2

Assegnazione di una funzione ai tasti funzione durante lo stato calcoli immediati 8-6

Impiego dei tasti funzione nello stato comandi 8-7

9. IMPIEGO DI UN VIDEO DISPLAY 9-1

Attivazione del video display 9-2

Informazioni visualizzate sullo schermo 9-2

Impaginazione 9-4

Istruzioni BASIC e comandi per il video display 9-4

DISP 9-5
ERASE 9-9
REVERSE 9-11

A. PROGRAMMI DI UTILITA' A-1

DCOPY A-3
DINIT A-9
FLCOPY A-13
FLPRINT A-19
LBCREATE A-23
LBEMPTY A-27
LBPROTECT A-31
LBRENAME A-35
LBSCRATCH A-39
LIBCOPY A-41
RESTRUCT A-47
VOLLABEL A-51

B. RIGENERAZIONE DEL SISTEMA OPERATIVO SU DCU O HDU B-1

Floppy disk richiesti B-1

Unità a disco richieste B-1

Procedura di generazione B-2

Note B-7

C. CARATTERI DI CONTROLLO C-1

D. MESSAGGI DEL SISTEMA P6066 D-1

Messaggi di avvertimento D-1

Messaggi informativi D-1

Messaggi di errore D-2

Messaggi di funzionamento anormale D-17

E. SET DI CARATTERI DEL SISTEMA P6066 E-1

F. OCCUPAZIONE DEI DATI IN MEMORIA PRINCIPALE E SU FLOPPY DISK F-1

G. CLASSIFICAZIONE FUNZIONALE DEI COMANDI E PROGRAMMI DI UTILITA' G-1

INDICE DELLE FIGURE

		Pag.
1-1	La configurazione minima del sistema P6066 con il sistema operativo residente su floppy disk	1-2
1-2	La configurazione minima del sistema P6066 con il sistema operativo residente su disco	1-3
1-3	La tastiera	1-5
1-4	Sezione BASIC/alfanumerica	1-6
1-5	Sezione di editing	1-8
1-6	Sezione algebrica	1-10
1-7	Sezione chiusura impostazioni	1-11
1-8	Sezione comandi	1-11
1-9	Sezione funzioni definibili	1-12
1-10	La console	1-13
1-11	Il display	1-15
1-12	Unità floppy disk	1-16
1-13	Divisione del floppy disk in tracce e settori	1-17
1-14	L'unità DCU	1-18
1-15	Divisione del disco (DCU) in tracce e settori	1-20
1-16	L'unità HDU	1-22
1-17	Divisione del disco (HDU) in tracce e settori	1-23
1-18	Sistema P6066 con la stampante integrata	1-25
1-19	Il sistema P6066 con il video	1-26

1-20	Unità floppy disk con 2 floppy disk	1-27
2-1	Parti componenti la stampante integrata	2-2
2-2	Avanzamento della carta nella stampante integrata	2-3
2-3	Inserimento del floppy disk in una unità monodisco	2-4
2-4	Inserimento del floppy disk in un'unità bidisco	2-5
2-5	Visualizzazione sul display di caratteri introdotti da tastiera o generati da programma (o sistema)	2-8
4-1	Campo di rappresentazione interna dei valori assegnati a variabili in singola precisione	4-3
4-2	Campo di rappresentazione dei valori numerici assegnati a variabili in doppia precisione	4-4
5-1	Buffer di display e relativo pointer	5-89
5-2	Buffer di stampa e relativo pointer	5-181
6-1	Campo della rappresentazione interna dei numeri	6-4
8-1	Tasti funzione	8-1
9-1	P6066 con video display Olivetti DSM 6680	9-1
9-2	Informazioni visualizzate sul video display	9-3

INDICE DELLE TABELLE

		Pag.
4-1	Caratteri speciali	4-2
4-2	Regole per la generazione dei nomi delle variabili	4-17
4-3	Funzioni numeriche di sistema	4-19
4-4	Funzioni stringa di sistema	4-22
5-1	Impiego della virgola e del punto e virgola con DISP	5-92
5-2	Buffer di stampa e relativo pointer	5-181
6-1	Funzioni numeriche di sistema	6-11
B-1	Risposte standard al messaggio UNIT? (1 DCU)	B-8
B-2	Risposte standard al messaggio UNIT? (2 DCU)	B-9
E-1	Set di caratteri del sistema P6066	E-3
F-1	Spazio allocato per i dati	F-1
G-1	Classificazione dei comandi e programmi di utilità	G-7

INTRODUZIONE

L'Olivetti P6066 è un minicomputer general-purpose, orientato verso applicazioni tecniche e scientifiche. Programmabile in BASIC, un linguaggio ad alto livello, interattivo, il P6066 mette a disposizione dell'utente anche non specializzato le prestazioni di un computer di notevole potenza.

Estremamente versatile, il P6066 può tracciare grafici, elaborare testi, eseguire calcoli immediati e funzionare come terminale intelligente. E' dotato di una veloce memoria interna ad accesso diretto e di una memoria di massa a dischi o floppy disk.

Il P6066, semplice e facile da usare, risponde a due esigenze fondamentali: ridurre i tempi necessari a risolvere un problema ed automatizzare al massimo tutte le operazioni. Per ottenere questi risultati, il P6066 offre una combinazione di hardware e software che può dirsi unica, in grado di aumentare la produttività, agevolando e semplificando al tempo stesso il lavoro.

Il nucleo del sistema è composto da una tastiera tecnologicamente avanzata, un display, una unità a floppy disk e una stampante integrata. Queste unità funzionano sotto il controllo di un sistema operativo che fornisce una potente versione del linguaggio BASIC, un insieme di comandi di sistema ed una serie di programmi di utilità.

Il linguaggio BASIC permette di rappresentare i numeri in virgola mobile, in singola e doppia precisione, di visualizzare e stampare testi con diversi formati predisposti dall'utente, di elaborare stringhe di caratteri e matrici, di elaborare file dati ad accesso diretto o sequenziale, di tracciare grafici, di collegare automaticamente i programmi tra loro con passaggio di dati comuni attraverso la memoria principale, di programmare una vasta gamma di periferiche e di richiamare ad eseguire moduli assembler.

I comandi di sistema permettono di introdurre programmi, correggerne gli errori e lanciarne l'esecuzione.

I programmi di utilità consentono una facile gestione delle librerie di programmi e dati.

Infine, comandi di sistema, programmi di utilità e programmi di utente, possono essere utilizzati per costruire delle procedure, che, una volta registrate, possono essere eseguite ogni qual volta l'utente lo richieda.

Questo manuale spiega come usare il P6066 per creare, eseguire, verificare, modificare e archiviare programmi e testi; come eseguire calcoli immediati e come utilizzare tutte le prestazioni del sistema. La sua struttura è tale da proporlo come strumento di lavoro indispensabile per una consultazione rapida su qualsiasi prestazione del sistema.

1. IL SISTEMA P6066

Questo capitolo descrive le configurazioni minime di sistema del P6066 e fornisce informazioni sulle possibili estensioni di sistema. Esso spiega come usare la tastiera, la console ed il display; inoltre fornisce informazioni sui dischi, floppy disk e stampanti. Una familiarità con i componenti del sistema può rendere più efficiente l'impiego del P6066 come strumento di soluzione dei problemi.

Configurazioni minime di sistema

Il P6066 offre due diverse configurazioni minime di sistema a seconda del supporto su cui risiede il sistema operativo: floppy disk o disco.

La figura 1-1 mostra la configurazione minima di sistema nel caso in cui il sistema operativo risieda su un floppy disk. Essa si compone delle seguenti parti:

- unità centrale di elaborazione
- memoria principale
- memoria a sola lettura
- governi di input-output
- tastiera e console
- display e segnalatore acustico
- unità a floppy disk (ad un trascinatore)

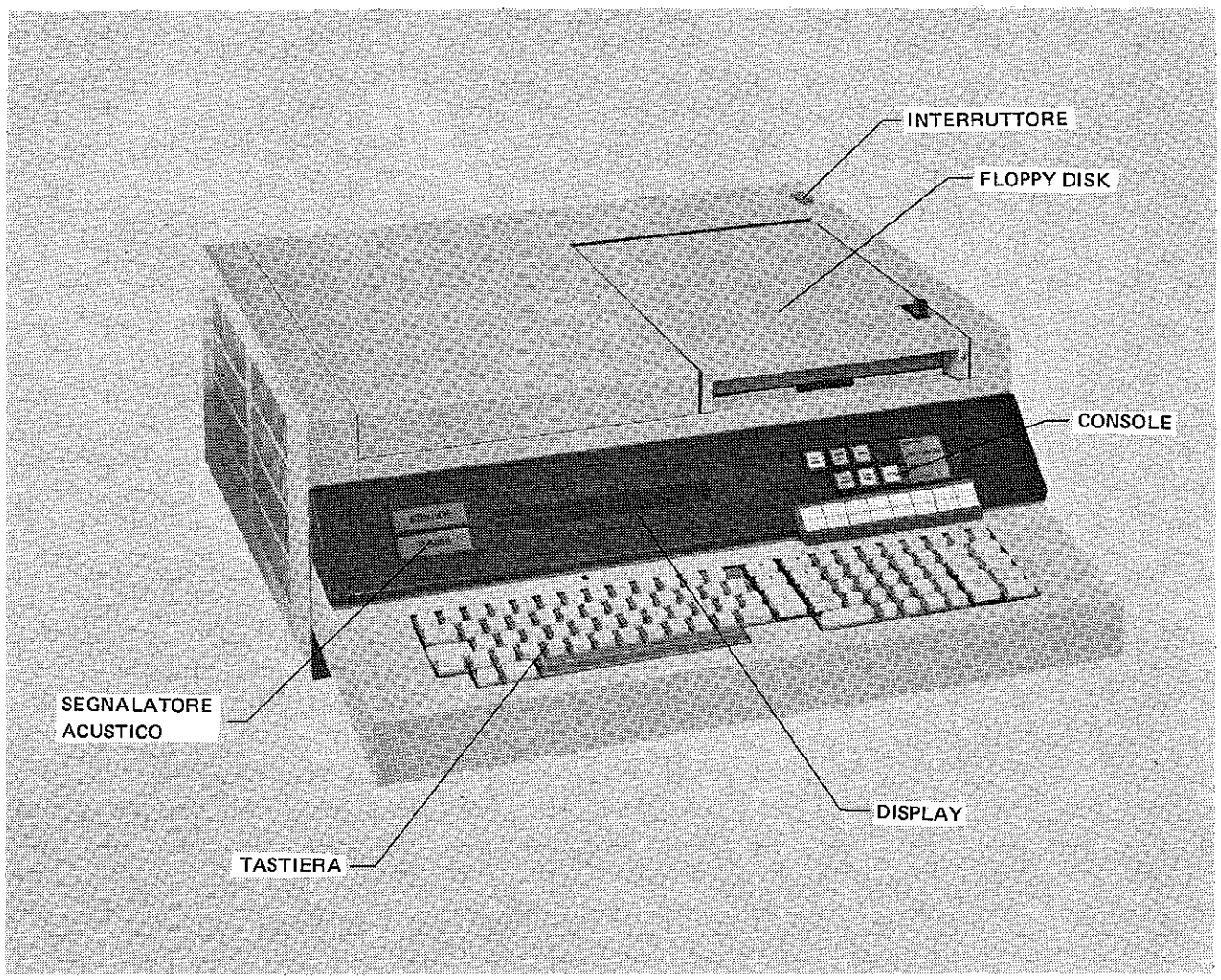


Figura 1-1 La configurazione minima del sistema P6066 con il sistema operativo residente su floppy disk

Nella figura 1-2 è illustrata la configurazione minima di sistema nel caso in cui il sistema operativo risieda su di un disco. Essa si compone delle parti elencate in precedenza e di un'unità a disco che può essere un HDU oppure un DCU.

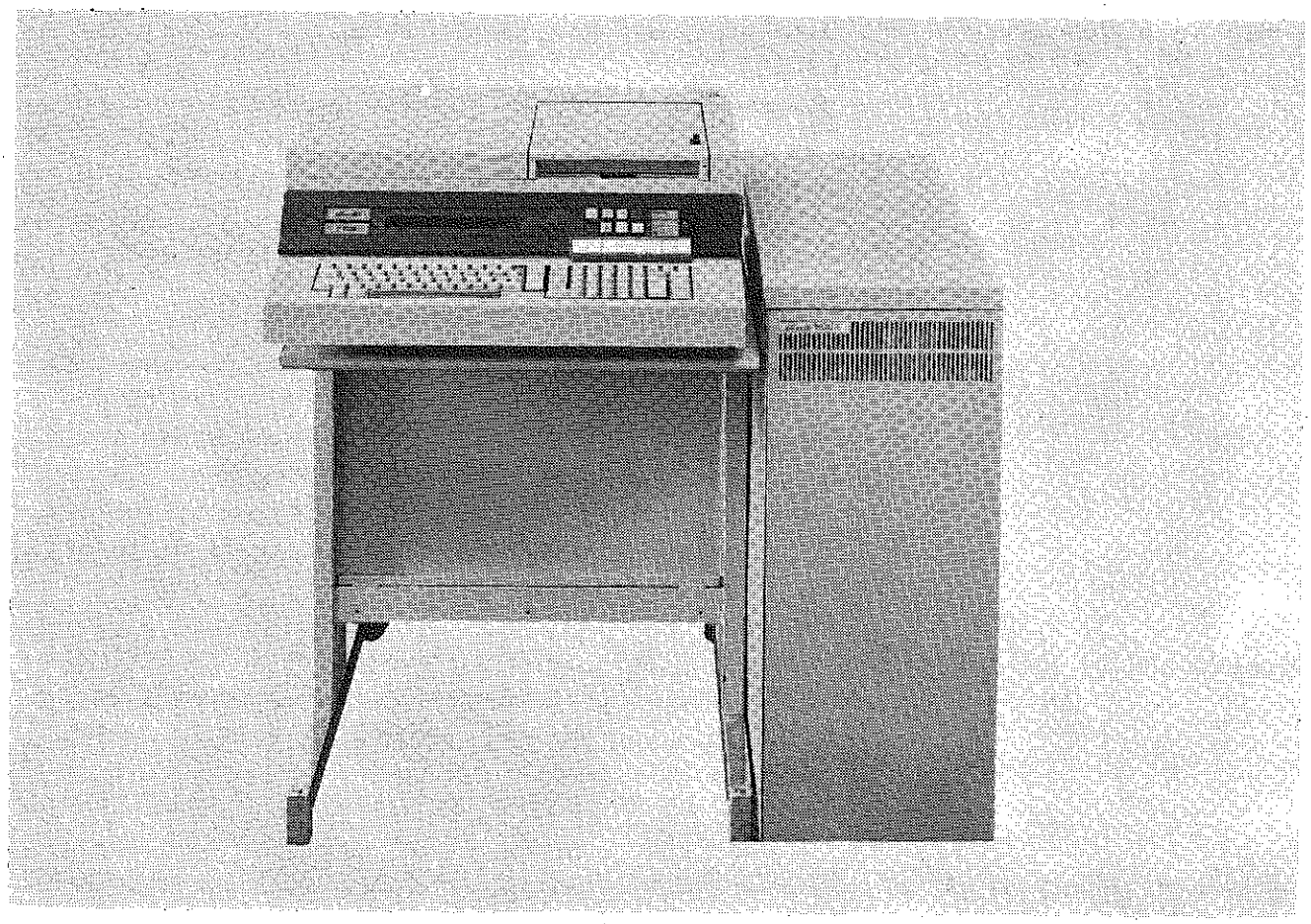


Figura 1-2 La configurazione minima del sistema P6066 con il sistema operativo residente su disco

Ognuna delle parti componenti le configurazioni suddette è descritta in dettaglio nei paragrafi che seguono.

Unità centrale di elaborazione

L'unità centrale di elaborazione, detta brevemente CPU dall'inglese Central Processing Unit, intrepreta ed esegue le istruzioni contenute nella memoria principale.

Essa dispone di 16 canali di I/O attraverso i quali vengono eseguiti i trasferimenti di informazioni da un supporto esterno in memoria principale (operazioni di input) e dalla memoria principale ad un supporto esterno (operazioni di output).

Memoria principale

La memoria principale, realizzata con circuiti integrati MOS, contiene:

- microprogrammi, che interpretano ed eseguono le

istruzioni del programma utente

- programmi del software di base (sistema operativo), che presiedono e controllano tutte le operazioni del sistema

- programmi utente, con i dati da elaborare ed i risultati delle elaborazioni

Possiamo quindi distinguere la memoria principale in due parti: una parte contenente il firmware (microprogrammi) ed il software di base, detta memoria di sistema, ed una parte contenente i programmi utente da eseguire, detta memoria utente. La memoria utente, nella configurazione base, ha una capacità di 16384 byte di 8 bit ciascuno.

Memoria a sola lettura

La memoria a sola lettura, brevemente ROM dall'inglese Read Only Memory, contiene i microprogrammi di inizializzazione che sono eseguiti quando il sistema è acceso o reinizializzato.

Governi di input/output

I governi di input/output controllano il trasferimento di informazioni da un supporto esterno alla memoria principale (input), o dalla memoria principale ad un supporto esterno (output). I governi di input/output si distinguono in governi che trasferiscono dati fra supporti esterni e memoria attraverso la CPU e governi che operano senza l'intervento dell' unità centrale (governi in DMA).

Appartengono al primo tipo:

- il governo di input da tastiera
- il governo di output su display
- il governo di input/output da/su floppy disk

Appartengono al secondo tipo:

- il governo DCC 6627 (per input/output da/su DCU)
- il governo HDC 6614 (per input/output da/su HDU)

Tastiera

La tastiera è composta da 96 tasti monostabili; con essa si comunicano al sistema dati, comandi ed istruzioni, attraverso un registro di transito (detto buffer) di 80 caratteri. La tastiera, vedi figura 1-3, è divisa nelle seguenti sezioni:

- sezione BASIC-alfanumerica
- sezione di editing
- sezione algebrica
- sezione di chiusura impostazioni
- sezioni comandi
- sezione funzione definibili

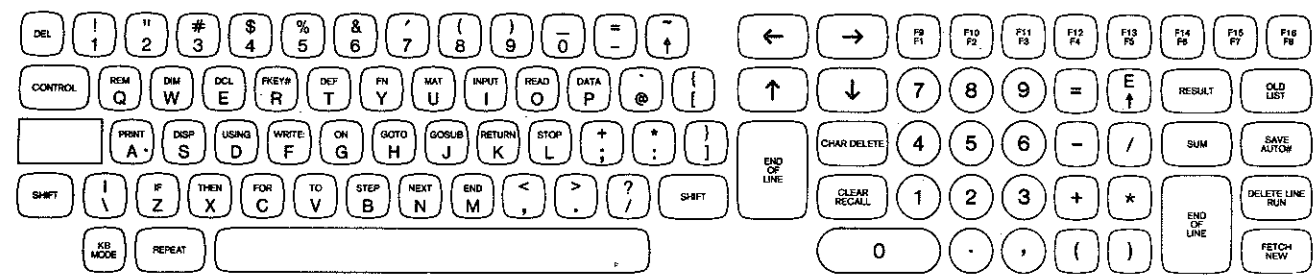


Figura 1-3 La tastiera

Sezione BASIC-alfanumerica: La sezione BASIC-alfanumerica è simile alla tastiera di una macchina per scrivere, le cifre numeriche sono sulla riga più in alto e la barra spaziatrice è nella riga più in basso. Questa sezione è stata progettata particolarmente per utenti del linguaggio BASIC. Invece di lettere maiuscole e minuscole, sui tasti sono impresse le lettere maiuscole e le parole chiave BASIC più usate. Quando si preme uno di questi tasti, viene introdotta una lettera maiuscola; quando si preme uno di questi tasti insieme ad uno dei due tasti [SHIFT] , si introduce una intera parola chiave BASIC.

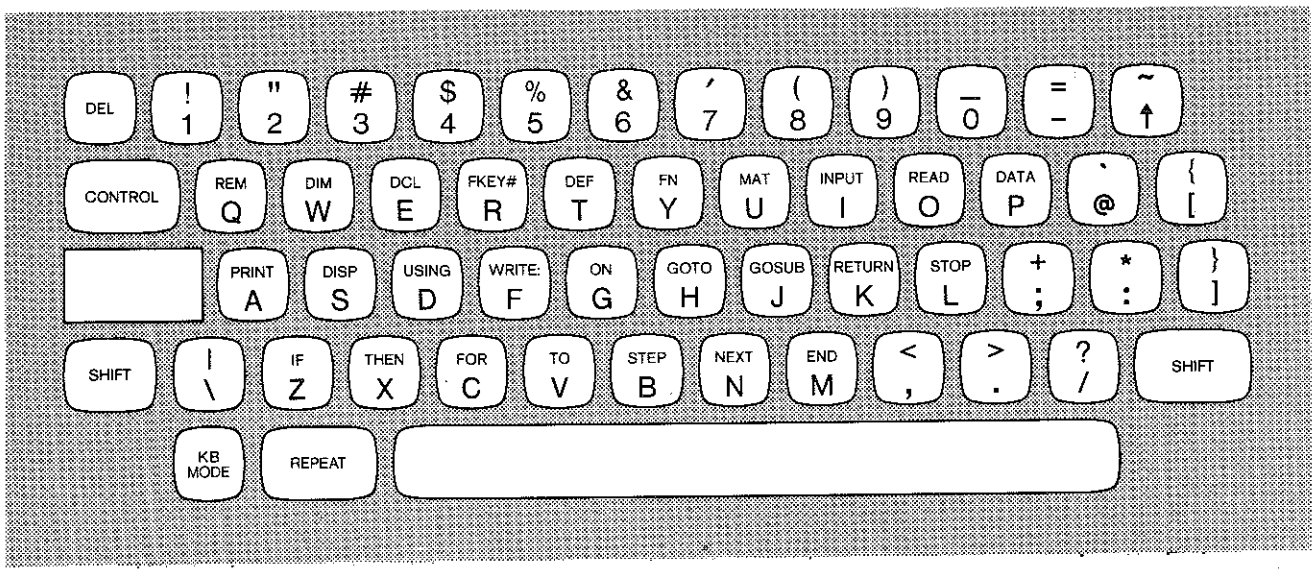


Figura 1-4 Sezione BASIC/alfanumerica

Con i tasti della sezione BASIC/alfanumerica si possono comporre e comunicare al sistema:

- istruzioni di programma es.
  1 2 0 SHIFT PRINT A B
- istruzioni di calcolo immediato es.
  1 2 3 · C O S (P I)
- dati numerici es. 4 . 5
- stringhe di caratteri es. V O L U M E
- comandi di sistema es. N E W
- comandi che richiamano ed eseguono programmi di utilità es. E X E D C O

I tasti della sezione BASIC/alfanumerica generano e trasmettono al sistema i 128 codici ISO che sono riportati nell'appendice E ed i codici relativi a 26 parole chiave di altrettante istruzioni BASIC.

La statiera alfanumerica è costituita dai seguenti tasti:

DEL — (DELETE) genera il codice ISO DEL che è operativo nel collegamento del sistema a periferiche esterne (vedi manuale "P6066 I/O con periferiche esterne". Quando questo codice è inviato in stampa od in display ad esso corrisponde il simbolo ⁂.

CONTROL — (CONTROL) è operativo nel collegamento del sistema a periferiche esterne (vedi manule "P6066 I/O con periferiche esterne") ed è premuto insieme ad un tasto della sezione BASIC/alfanumerica come indicato in appendice C.

SHIFT — (SHIFT) premuto insieme ad un tasto con due chiavi introduce la chiave superiore.

KB MODE — (KEYBOARD MODE) quando è premuto permette di utilizzare la sezione BASIC/alfanumerica come una macchina da scrivere: premendo SHIFT insieme ad un tasto con una parola chiave BASIC si introduce la lettera maiuscola indicata nella parte inferiore del tasto. In questo caso un indicatore luminoso posto sull'estremità sinistra della tastiera è acceso. Per uscire dal keyboard mode si deve premere di nuovo KB MODE.

LETTERE ALFABETICHE — Corrispondenti alle lettere maiuscole e minuscole dell'alfabeto inglese

CARATTERI NUMERICI — Le cifre da Ø a 9

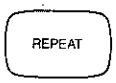
CARATTERI ALFANUMERICI — L'unione dei due insieme di caratteri suddetti

CARATTERI SPECIALI — Si possono generare i seguenti caratteri:

- le cifre da Ø a 9
- i segni di interpunzione: , . ; : ! ? " ' (apostrofo)
- le parentesi: (,) , [,] , { , }
- l'accento grave: `
- i segni di confronto o assegnazione:
  <> o >< non uguale
  => o >= maggiore di o uguale a
  =< o <= minore di o uguale a
  > maggiore di
  < minore di
  = uguale a o assegnazione

- i segni aritmetici: + , -, *, /, ↑

- i caratteri: #, $, % , & , _ , - ,~ , @, | ,\



(REPEAT) quando è premuto insieme ad un altro tasto viene ripetuta la generazione dello stesso codice.

BARRA SPAZIATRICE

Genera un codice che indica assenza di carattere grafico.

Sezione di editing: La sezione di editing è costituita dai 7 tasti indicati nella figura 1-5.

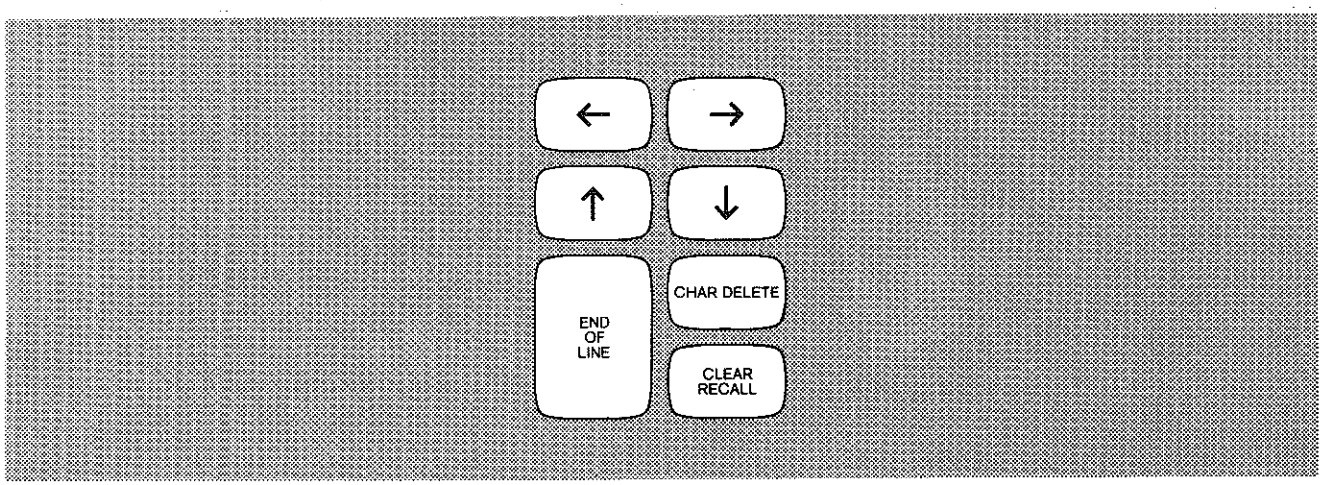
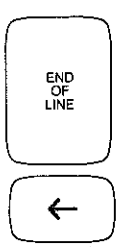


Figura 1-5 Sezione di editing



(END OF LINE) completa l'introduzione da tastiera

L'effetto prodotto premendo questo tasto dipende dalla posizione del pointer sul display (vedi cap. 1, par. "Il display"):

- se il pointer è in una posizione compresa tra 1 e 31, il pointer si sposta di una posizione sulla sinistra

- se il pointer è in posizione 32, ma i caratteri visualizzati non sono l'inizio del testo, tutti i caratteri sul display si spostano di una posizione a destra; il pointer rimane nella posizione 32

- se il pointer è all'inizio del display, il tasto non provoca alcun effetto

Premendo (SHIFT) insieme a questo tasto, l'inizio del testo viene visualizzato a partire dalla seconda posizione ed il pointer è in prima posizione.

[→] L'effetto prodotto premendo questo tasto dipende dalla posizione del pointer sul display:

- se il pointer è in una posizione compresa tra 1 e 31 il pointer si sposta di una posizione verso destra

- se il pointer è in posizione 32, ma non alla fine del testo, tutti i caratteri sul display si spostano di una posizione a sinistra; il pointer rimane nella stessa posizione

- se il pointer è alla fine del testo, premendo il tasto non si ha alcun effetto

Premendo [SHIFT] insieme a questo tasto il pointer viene posizionato alla fine del testo (se il testo ha più di 31 caratteri vengono visualizzati gli ultimi 31 caratteri).

[CHAR DELETE] (CHARACTER DELETE) quando è premuto cancella il carattere che è visualizzato sul display alla sinistra della freccia luminosa (pointer) e sposta di una posizione verso sinistra sia il pointer che gli eventuali caratteri alla destra di quest'ultimo.

[↑] Permette di viasualizzare sul display l'istruzione di programma o la linea di testo che precede, secondo il numero di linea, quella presente sul display prima che il suddetto tasto sia premuto. Se la linea visualizzata sul display è la prima di un programma o di un file testo, premendo il suddetto tasto appare sul display l'ultima linea del programma o del file testo.

[↓] Permette di visualizzare sul display l'istruzione di programma o la linea di testo che segue, secondo il numero di linea, quella presente sul display prima che il suddetto tasto sia premuto. Se la linea visualizzata sul display è l'ultima di un programma o di un file testo, premendo il tasto suddetto appare sul display la prima linea del programma o del file testo.

[CLEAR RECALL] (CLEAR/RECALL)

- se premuto insieme al tasto [SHIFT] cancella tutti i caratteri introdotti prima di premere il tasto [END OF LINE] e sblocca la tastiera se era stata bloccata dopo una segnalazione di errore

- se premuto senza premere il tasto [SHIFT] visualizza alternativamente i caratteri introdotti da tastiera o l'ultimo messaggio apparso sul display.

Sezione algebrica: La sezione algebrica è costituita dai 21 tasti evidenziati in figura 1-6.

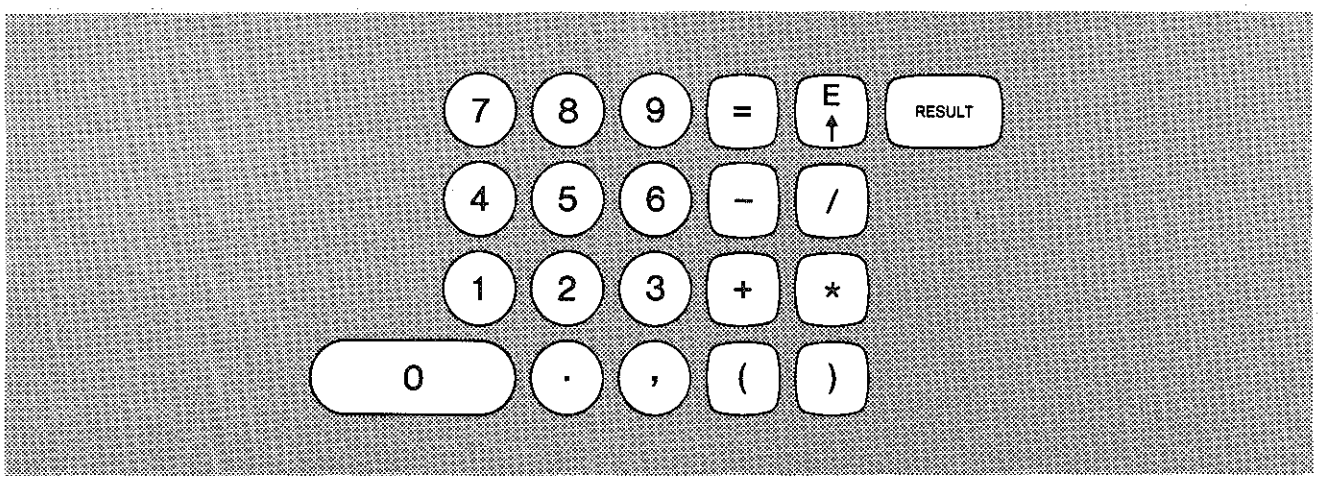


Figura 1-6 Sezione algebrica

TASTI NUMERICI — Quando sono premuti generano le cifre da Ø a 9.

[.] — Quando è premuto comunica al sistema che le cifre che verrano impostate successivamente sono da considerarsi decimali.

[E ↑] — Questo tasto è usato per introdurre numeri nel formato esponenziale. Se si usa E, il numero introdotto è elevato ad una potenza di 10. Se si usa ↑ il numero introdotto è elevato ad una potenza specificata dal programmatore. Per esempio, premendo nell'ordine questi tasti: [1] [2] [SHIFT] [E] [5] il numero fornito al sistema è 12×10^5. Se si premono nell'ordine questi tasti: [1] [2] [↑] [5] il numero fornito al sistema è 12^5.

TASTI ARITMETICI — Quando sono premuti richiedono al sistema di eseguire un operazione aritmetica di:

- addizione [+]
- sottrazione [-]
- moltiplicazione [*]
- divisione [/]
- elevazione a potenza [↑]
- assegnazione [=]
- [(] e [)] sono utilizzati per dare una priorità univoca alla esecuzione delle operazioni artimetiche contenute in una espressione.

Nota: Tutti i tasti precedentemente descritti sono riportati anche nella sezione BASIC/alfanumerica.

(RESULT) viene usato quando si comanda al sistema di eseguire calcoli immediati. Sul display viene visualizzato Φ . (Vedi "Esecuzione di calcoli immediati", capitolo 6).

Sezione chiusura impostazioni: La sezione "Chiusura impostazioni" è composta dai due tasti di figura 1-7:

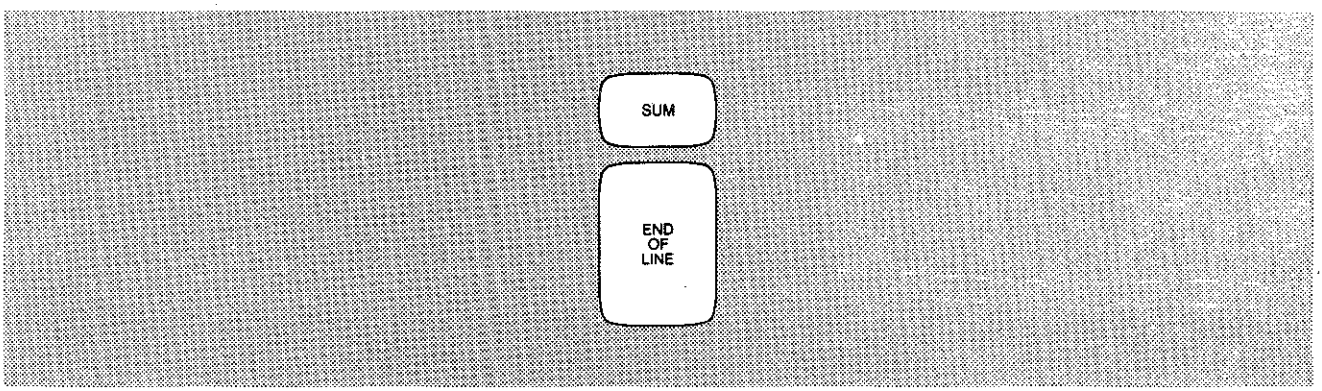
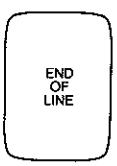
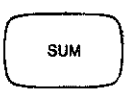


Figura 1-7 Sezione chiusura impostazioni



(END OF LINE) è la duplicazione del tasto descritto nella sezione di Editing.



(SUM) è usato per totalizzare l'esecuzione di calcoli immediati (vedi capitolo 6).

Sezione comandi: La sezione "Comandi" è costituita dai 4 tasti indicati in figura 1-8.

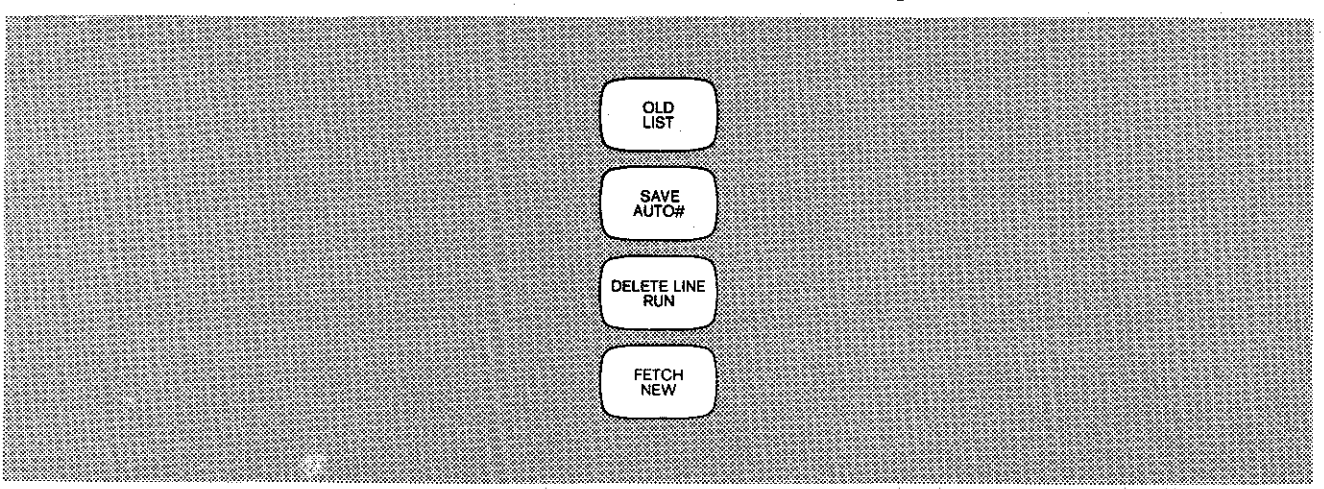


Figura 1-8 Sezione comandi

OLD LIST

(OLD/LIST) con SHIFT comunica al sistema il comando OLD che carica in memoria principale un file memorizzato in una libreria. Senza SHIFT comunica al sistema il comando LIST che stampa il contenuto del file o di parte del file presente in memoria principale.

SAVE AUTO#

(SAVE/AUTO #) con SHIFT comunica al sistema il comando SAVE che memorizza in una libreria il file presente in memoria principale. Senza SHIFT comunica al sistema il comando AUTO# che genera la numerazione automatica dei numeri di linea durante la creazione di un programma o di un file testo.

DELETE LINE RUN

(DELETE LINE/RUN) con SHIFT comunica al sistema il comando DELETE LINE che cancella in memoria principale la linea o le linee specificate nel comando. Senza SHIFT comunica al sistema il comando RUN che esegue un programma presente in memoria principale.

FETCH NEW

(FETCH/NEW) con SHIFT comunica al sistema il comando FETCH che invia sul display e nel buffer di tastiera una linea di programma o di file testo presente in memoria principale. Senza SHIFT comunica al sistema il comando NEW che specifica che le linee introdotte dopo sono istruzioni di programma.

<u>Sezione funzioni definibili</u>: E' composta dagli 8 tasti di figura 1-9:

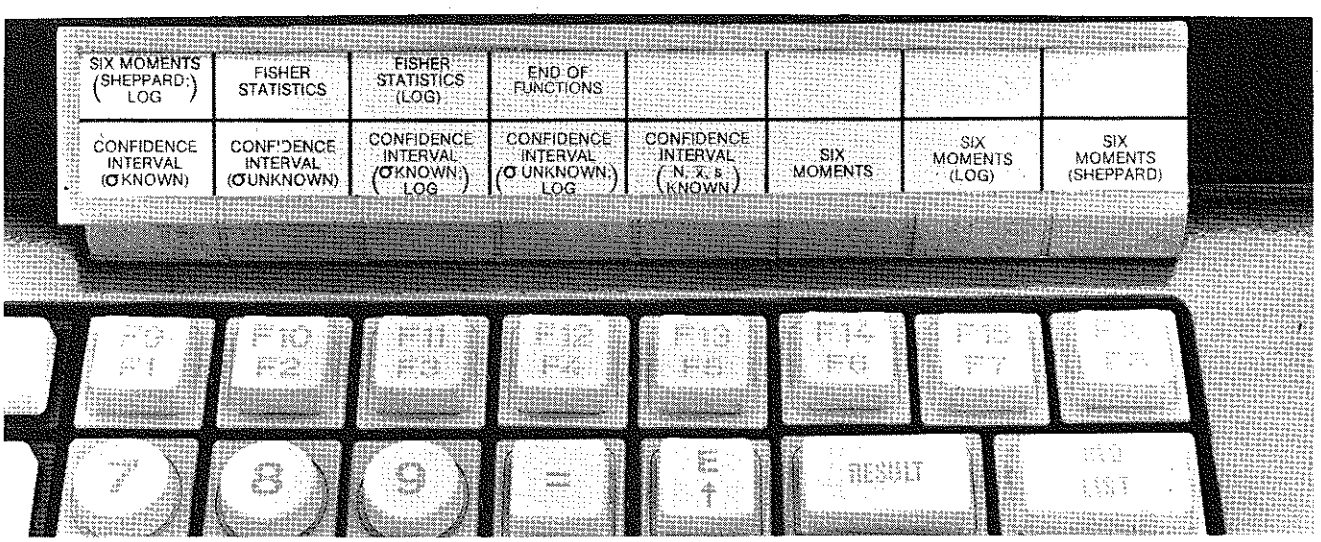


Figura 1-9 Sezione funzioni definibili

Si possono associare alle 16 funzioni F1 ÷ F16 indicate in figura delle sequenze di caratteri prestabilite che vengono comunicate al sistema ogni volta che si preme il tasto corrispondente. Ad ogni tasto sono associate due funzioni e quella indicata nella parte superiore è abilitata con il tasto [SHIFT] (vedi capitolo 8 per l'impiego dei tasti funzione).

La console

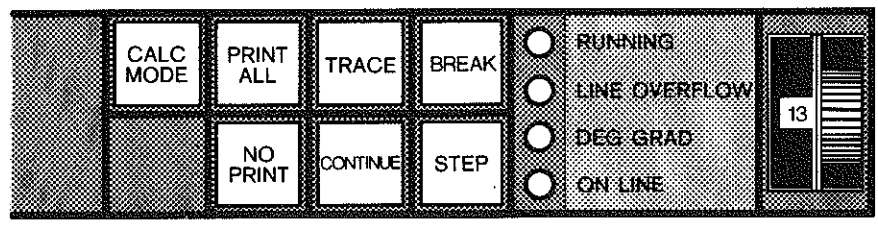


Figura 1-10 La console

La console è costituita da 7 tasti, 4 indicatori luminosi ed un dispositivo d'impostazione dei decimali, come si vede in figura 1-10. I 7 tasti hanno incorporata una lampadina che si accende quando la funzione associata è attiva.

(CALCULATOR MODE) premendo il tasto si pone il sistema nello stato di esecuzione di calcoli immediati. Quando il sistema è nello stato di esecuzione di calcoli immediati la luce incorporata è accesa.

(PRINT ALL) premendo il tasto tutti i testi che appaiono sul display sono stampati. Quando la funzione suddetta è attiva la luce incorporata è accesa. Per disattivare la funzione PRINT ALL basta ripremere il tasto.

(NO PRINT) premendo il tasto vengono inibite tutte le operazioni di stampa riferite alla stampante integrata. Quando la suddetta funzione è attiva la luce incorporata è accesa. Per disattivare la funzione NO PRINT basta premere di nuovo il tasto.

(BREAK) premendo il tasto si termina l'esecuzione di un programma, del comando LIST o del comando CATALOG oppure si interrompe l'esecuzione di una procedura catalogata (vedi comando PROCEDURE nel capitolo 3). La luce rimane accesa finchè la funzione BREAK è attiva.

I seguenti 3 tasti sono utilizzati in debugging.

(TRACE) premendo il tasto vengono stampati i numeri di linea delle istruzioni eseguibili di un programma durante la sua esecuzione. I numeri di linea sono stampati secondo l'ordine di esecuzione. Quando la suddetta funzione è attiva la luce incorporata è accesa. Per disattivare la funzione TRACE si deve ripremere il tasto.

(STEP) premendo il tasto si interrompe l'esecuzione di un programma e si possono verificare i valori correnti delle variabili. Ogni volta che il tasto è premuto successivamente il sistema esegue una istruzione. Quando la funzione suddetta è attiva la luce incorporata è accesa. (Vedi capitolo 7 per l'impiego del tasto per il debugging di un programma).

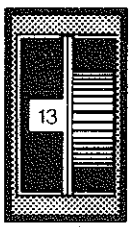
(CONTINUE) premendo il tasto viene ripresa l'esecuzione del programma interrotto. Quando la funzione suddetta è attiva la luce incorporata è accesa.

Se lampeggia indica che il sistema sta eseguendo una o più operazioni; oppure sta segnalando un errore di sintassi (non sono abilitati i tasti END OF LINE e SUM); se fissa indica che il sistema è in attesa di una introduzione da tastiera.

Se accesa indica che sono già stati introdotti 80 caratteri nel buffer di tastiera.

Si accende quando si introduce il comando SDEG o SGRAD (esecuzione di calcoli immediati); indica che i valori degli angoli sono misurati in gradi sessagesimali o centesimali.

Se accesa indica che la tastiera funziona come periferica esterna (ved. manuale "P6066 I/O con periferiche esterne").



(INDICATORE DEI DECIMALI) permette di predisporre il formato per la stampa e la visualizzazione dei numeri quando si eseguono calcoli immediati (vedi il capitolo 6).

Il display

Il display permette la visualizzazione di 32 caratteri compresi nel set dell'appendice E.

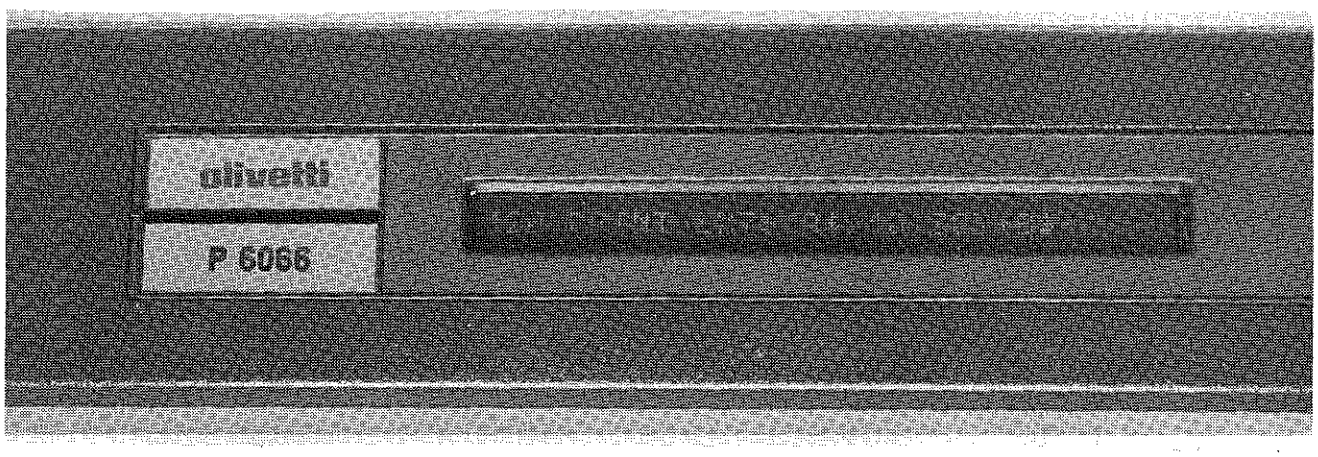


Figura 1-11 Il display

Sul display appaiono:

- linee generate da tastiera
- messaggi da programma
- messaggi da sistema
- segnalazioni di errore
- richieste di dati da tastiera (istruzioni INPUT, MAT INPUT, RKB)
- dati introdotti da tastiera
- caratteri introdotti con i tasti funzione

Quando si introduce da tastiera una linea, sul display compare anche una freccia luminosa, detta pointer, che indica in quale posizione della linea si può introdurre un nuovo carattere. Quando il display è predisposto a visualizzare i caratteri introdotti da tastiera, il pointer è nella prima posizione di sinistra e si sposta di una posizione sulla destra man mano che si introducono i caratteri; se si introducono più di 31 ca-

ratteri sul display sono visibili gli ultimi 31 caratteri introdotti: premendo [←] il pointer si sposta di una posizione a sinistra e sul display appaiono gli ultimi 32 caratteri introdotti.

Unità floppy disk

L'unità floppy disk è una unità con testina di lettura/scrittura mobile a dischi intercambiabili contenuta nel P6066 come si vede in fig. 1-12; essa comprende un trascinatore per inserire in esso un floppy disk.

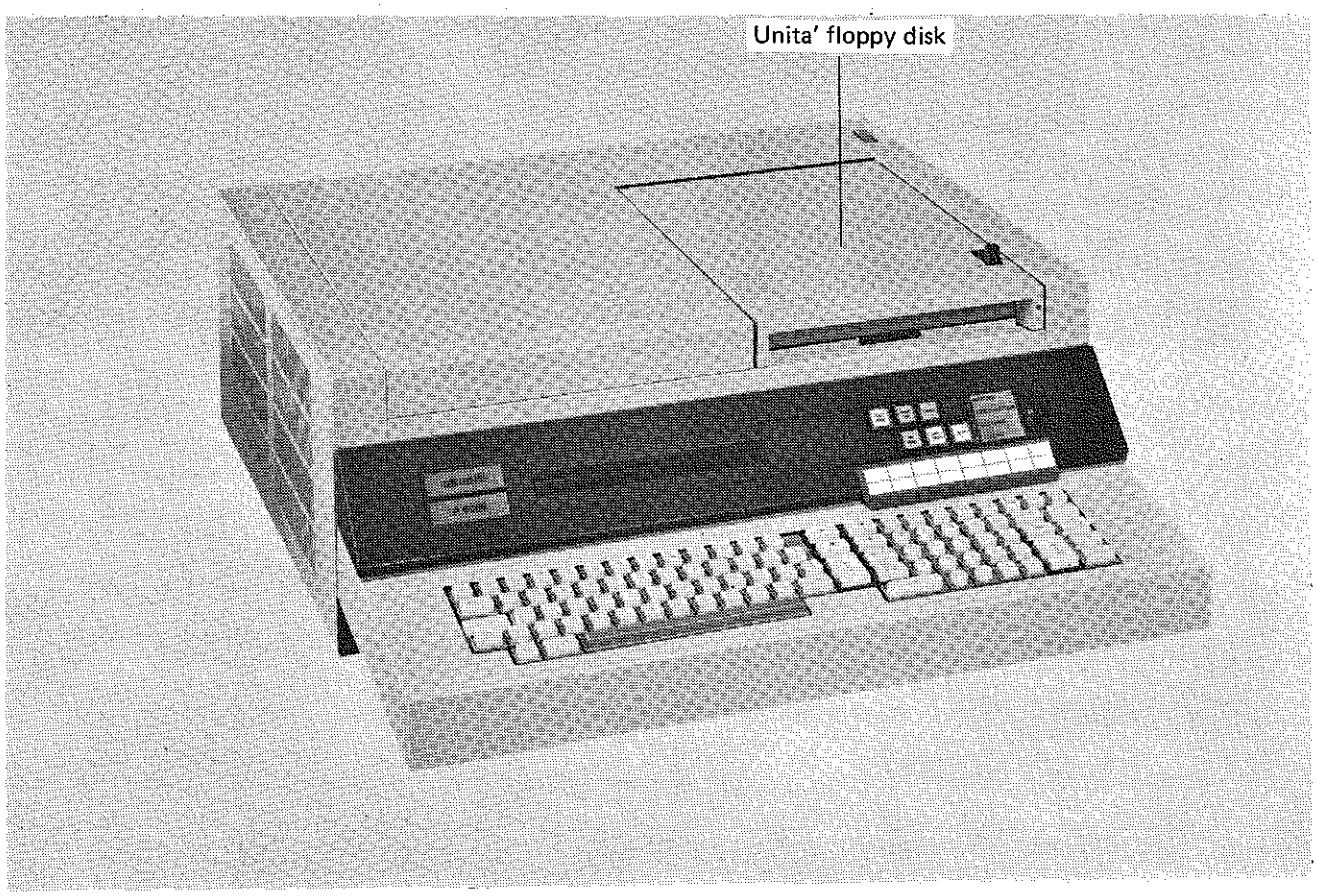


Figura 1-12 Unità floppy disk

Il floppy disk è un supporto di mylar, ricoperto di materiale magnetico ed inserito in una busta di carta dentro la quale esso è libero di ruotare. Il floppy disk è una memoria di massa ad accesso diretto, con tempo di accesso di 333 msec, che, ruotando ad una certa velocità, permette di leggere o registrare su di esso delle informazioni con una velocità di trasferimento di 250K bit/sec.; il floppy disk è inizializzato prima di essere spedito all'utente. Le informazioni sono memorizzate su sezioni circolari del disco (tracce). Ogni disco ha 77 tracce di cui 4

(tracce Ø, 74, 75 e 76) sono utilizzate in modo standard da qualunque sistema (vedi fig. 1-13). Ogni traccia è divisa in 26 settori di 128 byte ciascuno. La capacità del disco è di 240.640 byte. L'utente può conoscere lo spazio disponibile per la registrazione di informazioni sul disco utilizzando il comando di sistema SPACE (vedi capitolo 3).

Nei sistemi con il sistema operativo residente su floppy disk deve essere usato un floppy disk sistema. Il floppy disk sistema contiene microprogrammi firmware e moduli di sistema operativo. Nella parte rimanente l'utente può registrare dati e programmi oppure programmi applicativi forniti dalla Olivetti Spa.
Nei sistemi con il sistema operativo residente su disco si deve usare un floppy disk bootstrap. Il floppy disk bootstrap contiene programmi necessari per eseguire l'inizializzazione del sistema.

Dopo aver inizializzato il sistema si può sostituire il floppy disk bootstrap con un floppy disk utente sul quale si possono registrare programmi e dati.

Traccia 0 usata come traccia indice
Traccia 76 riservata
Traccia 74 e 75 in alternativa

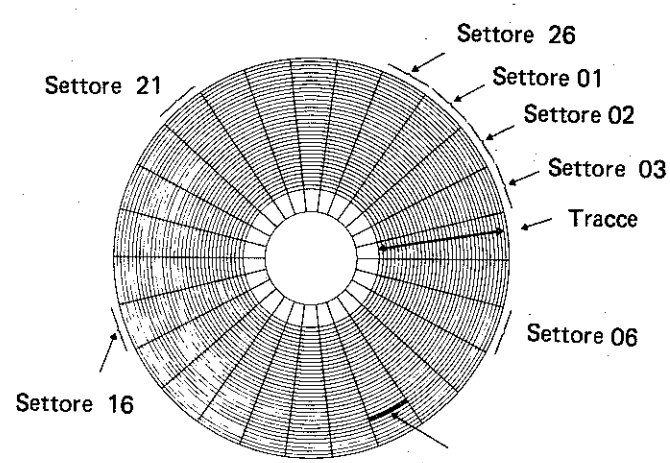


Figura 1-13 Divisione del floppy disk in tracce e settori

Segnalatore acustico

Come indicato nella figura 1-1 il sistema è fornito di un segnalatore acustico che emette un suono ogni qualvolta l'operatore introduce una linea non accettata dal sistema: ad esempio se si completa l'introduzione di una linea premendo EOL o SUM mentre la luce di con-

sole RUNNING lampeggia, oppure si introducono più di 80 caratteri prima di premere [END OF LINE] . Anche nel caso in cui si premono contemporaneamente due tasti viene emesso un suono e la digitazione è ignorata.

Interruttore

L'interruttore indicato nella figura 1-1, se è premuto dalla parte ON, accende il sistema, mentre, se è premuto dalla parte OFF, spegne il sistema.

Unità a disco

Il P6066 con il sistema operativo residente su disco può utilizzare uno di due tipi di unità a disco e precisamente:

DCU, dal nome inglese Disk Cartridge Unit
HDU, dal nome inglese Hard Disk Unit

L'unità DCU è illustrata in figura 1-14.

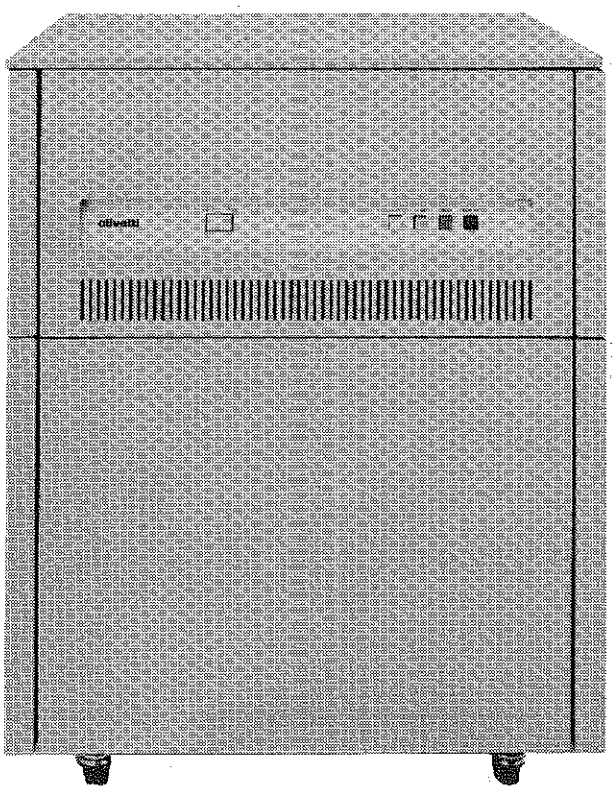


Figura 1-14 L'unità DCU

L'unità DCU utilizza come supporto di memorizzazione delle informazioni (programmi o dati) un disco con due superfici opposte ricoperte di materiale magnetico.

Nell'unità vi devono essere due dischi: uno fisso, situato nella parte inferiore dell'unità, ed uno rimovibile, situato nella parte superiore dell'unità. Si noti che è bene avere sempre inserito nell'unità DCU un disco rimovibile, anche se non si intende utilizzarlo in operazioni di I/O, perchè, in questo modo, l'interno dell'unità è protetto dalla polvere.

La superficie di un disco è divisa in sezioni circolari, dette tracce, che sono divise in settori circolari. Ogni coppia di tracce sulle due superfici opposte aventi lo stesso raggio è detta cilindro.

In figura 1-15 è indicata la struttura del disco sulle due superfici.

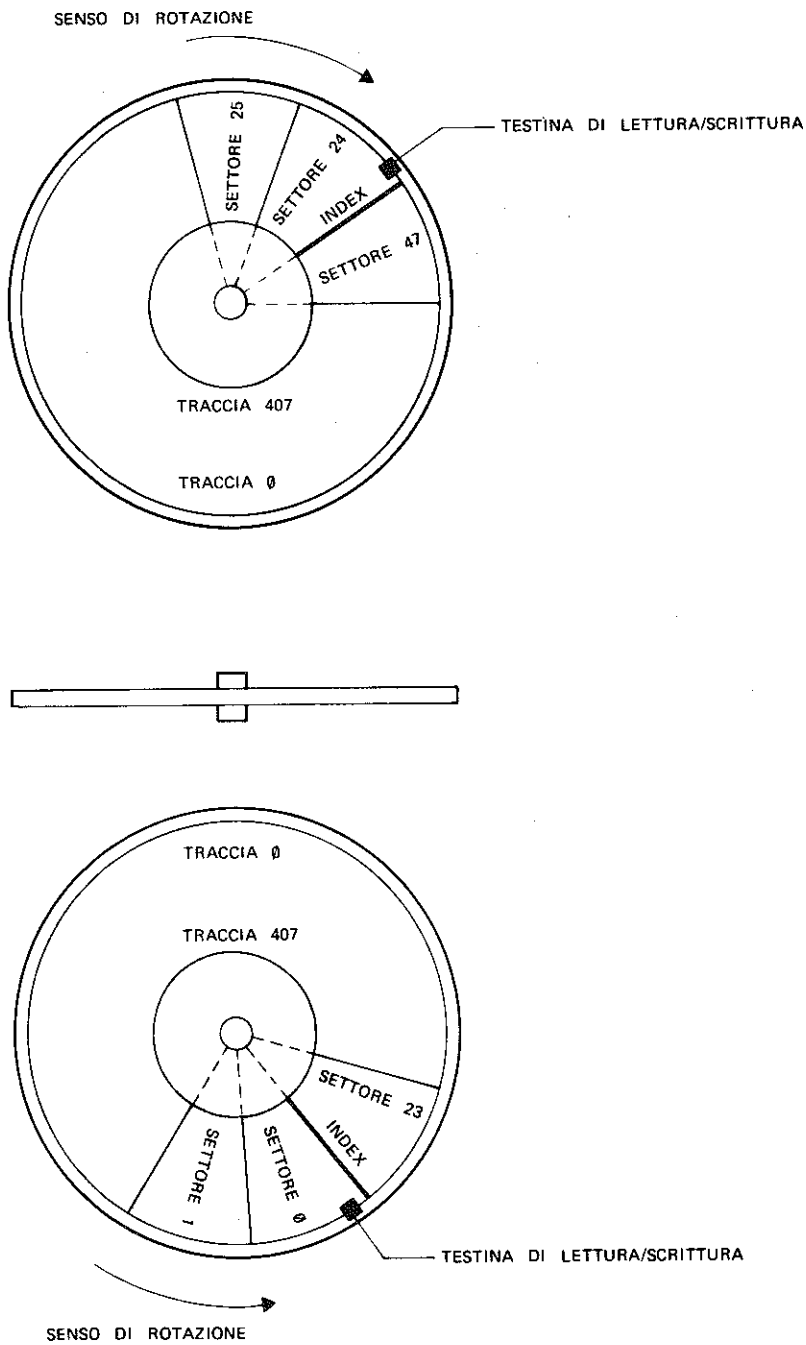


Figura 1-15 Divisione del disco (DCU) in tracce e settori

La freccia indica il senso di rotazione del disco rispetto alla testina di lettura/scrittura. Come si vede il disco è composto da:

- 408 cilindri, 6 di riserva e 2 di servizio

- 816 tracce (due per cilindro), 14 di riserva e 4 di servizio

- 19.584 settori (48 per cilindro)

La capacità del disco diponibile per l'utente è così distribuita:

- capacità del disco	4.915.200 byte
- capacità del cilindro	12.888 byte
- capacità della traccia	6.144 byte
- capacità del settore	256 byte

dove un byte è costituito da 8 bit.

L'unità DCU fornisce le seguenti prestazioni:

- tempo di posizionamento delle testine t_P

. tra tracce adiacenti	7,5 msec
. medio	35 msec
. massimo	60 msec

- tempo di latenza medio t_L = 12,5 msec
  (tempo di mezza rotazione)

- tempo di trasferimento di un settore t_T=0,819 msec

- tempo di attesa del settore successivo T_A =0,172 msec

Il tempo di esecuzione di una istruzione di trasferimento di n settori T_E è dato da:

$$t_E=t_P+t_L+n*t_T+(n-1)*t_A$$

Il sistema P6066 può essere collegato a due unità DCU per cui si possono avere fino a 4 dischi on line per un totale di capacità di 19.660.800 byte.

Per ulteriori informazioni sull'unità DCU si veda il manuale "DCU 7910 Memoria ausiliaria a dischi - Manuale generale", codice 3945350 Q.

Vi sono tre tipi di unità HDU:

- HDU 2102
- HDU 2105
- HDU 2110.

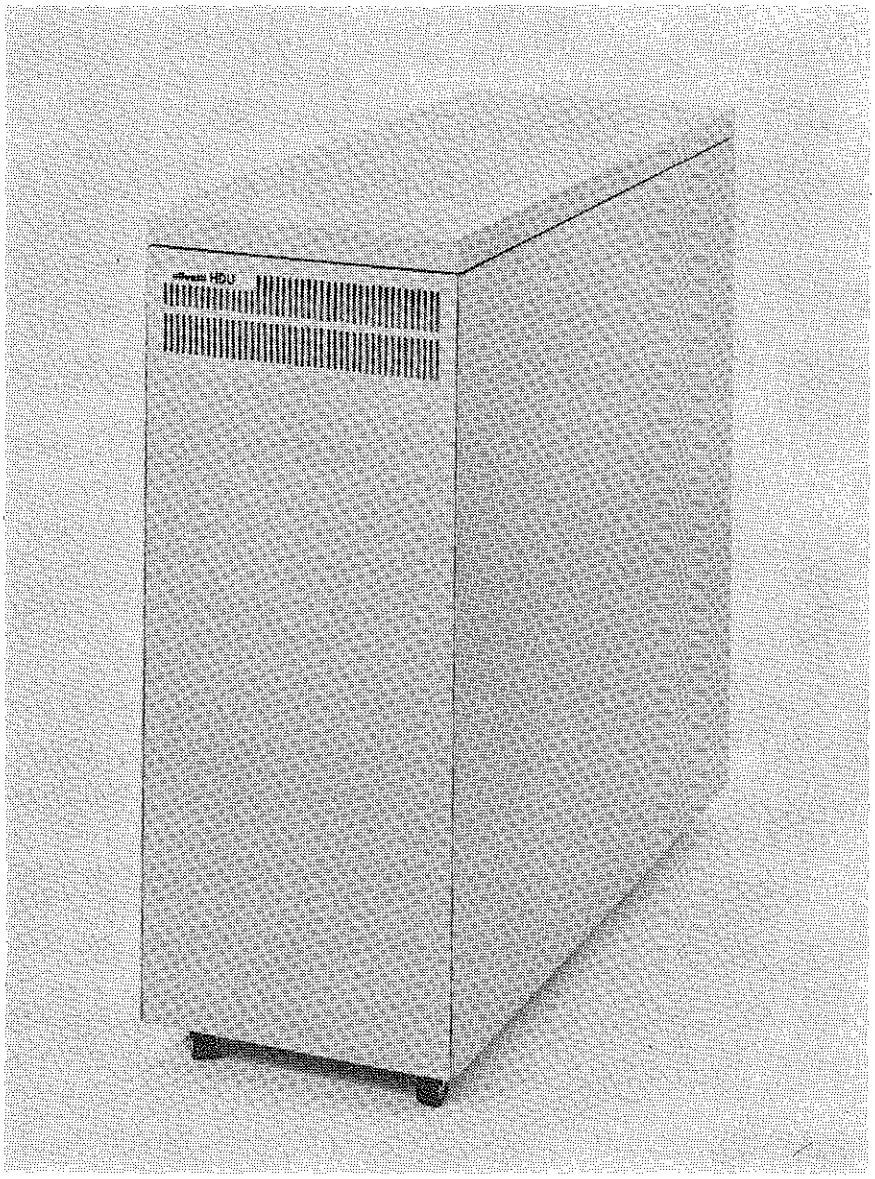


Figura 1-16 L'unità HDU

L'unità HDU 2102 dispone di un disco fisso con una sola superficie registrabile. L'unità HDU 2105 dispone di un disco fisso con due superfici registrabili. Infine, L'unità HDU 2110 dispone di due dischi fissi, entrambi con due superfici registrabili.

La superficie di un disco è divisa in sezioni circolari, dette tracce, che sono divise in settori circolari come indicato in figura 1-17.

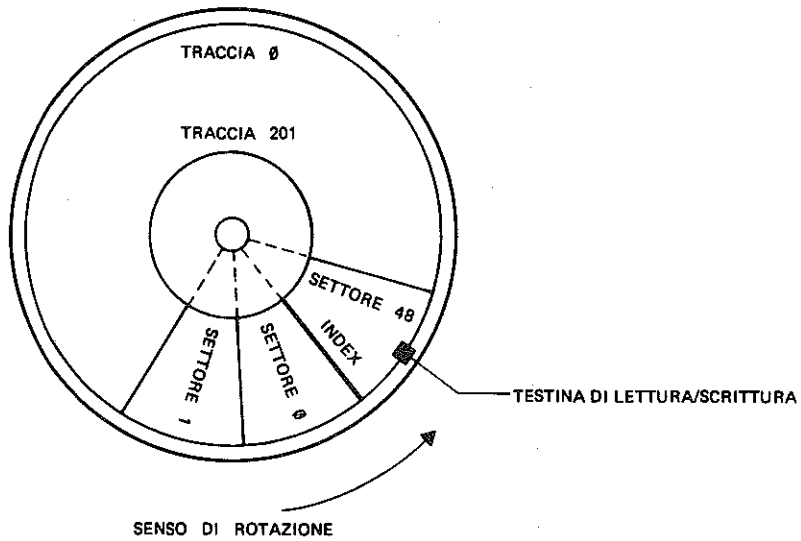


Figura 1-17 Divisione del disco (HDU) in tracce e settori

La freccia indica il senso di rotazione del disco rispetto alla testina di lettura/scrittura; come si vede la superficie è divisa in:

- 202 tracce, 2 di servizio
- 49 settori per traccia, 1 di riserva

La capacità disponibile per l'utente è così distribuita:

- capacità della superficie 2.457.600 byte
- capacità della traccia 12.288 byte
- capacità del settore 256 byte

Nelle unità HDU 2105 ed HDU 2110 tracce aventi lo stesso raggio costituiscono un cilindro per cui entrambe avranno 200 cilindri utente così composti: cilindro HDU 2105 di 2 tracce, quindi 96 settori utente; cilindro HDU 2110 di 4 tracce (due per disco), quindi 192 settori disponibili per l'utente.

La capacità utente di memoria di massa disponibile per il collegamento con il P6066 è per le tre unità HDU la seguente:

- HDU 2102 capacità di 2.457.600 byte

- HDU 2105 capacità di 4.915.200 byte
- HDU 2110 capacità di 9.830.400 byte

Le unità HDU forniscono le seguenti prestazioni:

- tempo di posizionamento delle testine t_P

  . tra tracce adiacenti 4 msec
  . medio 65 msec
  . massimo 164 msec

- tempo di latenza medio t_L = 20 msec

- tempo di trasferimento di un settore t_L = 0,654 msec

- tempo di attesa del settore successivo t_A = 0,166 msec

Il tempo di esecuzione di una istruzione di trasferimento di n settori t_E è dato da:

$$t_E = t_P + t_L + n * t_T + (n-1) * t_A$$

Il sistema P6066 può essere collegato a due unità HDU per cui, nel caso in cui si impiegano due unità HDU 2110, si possono avere fino a quattro dischi on-line per un totale di capacità di 19.660.800 byte.

L'unità non è dotata di una console operativa per cui è utilizzata come scatola chiusa ed è collegata al canale cavi del sistema P6066 per ricevere la tensione di rete. Per ulteriori informazioni si veda il manuale "HDU Memoria Ausiliaria a Dischi - Manuale Generale", codice 3942160 Y.

Configurazione estese

Partendo dalle configurazioni minime di sistema precedentemente descritte si possono realizzare configurazioni più estese che permettono di utilizzare numerose periferiche di input/output o di collegare il P6066 attraverso linea a computer remoti

La memoria utente può esser estesa dal valore minimo di capacità di 16K byte, presente nella configurazione base, fino ad un valore massimo di 48K byte.

Opzionalmente la macchina può includere una stampante integrata ed un secondo trascinatore per floppy disk.

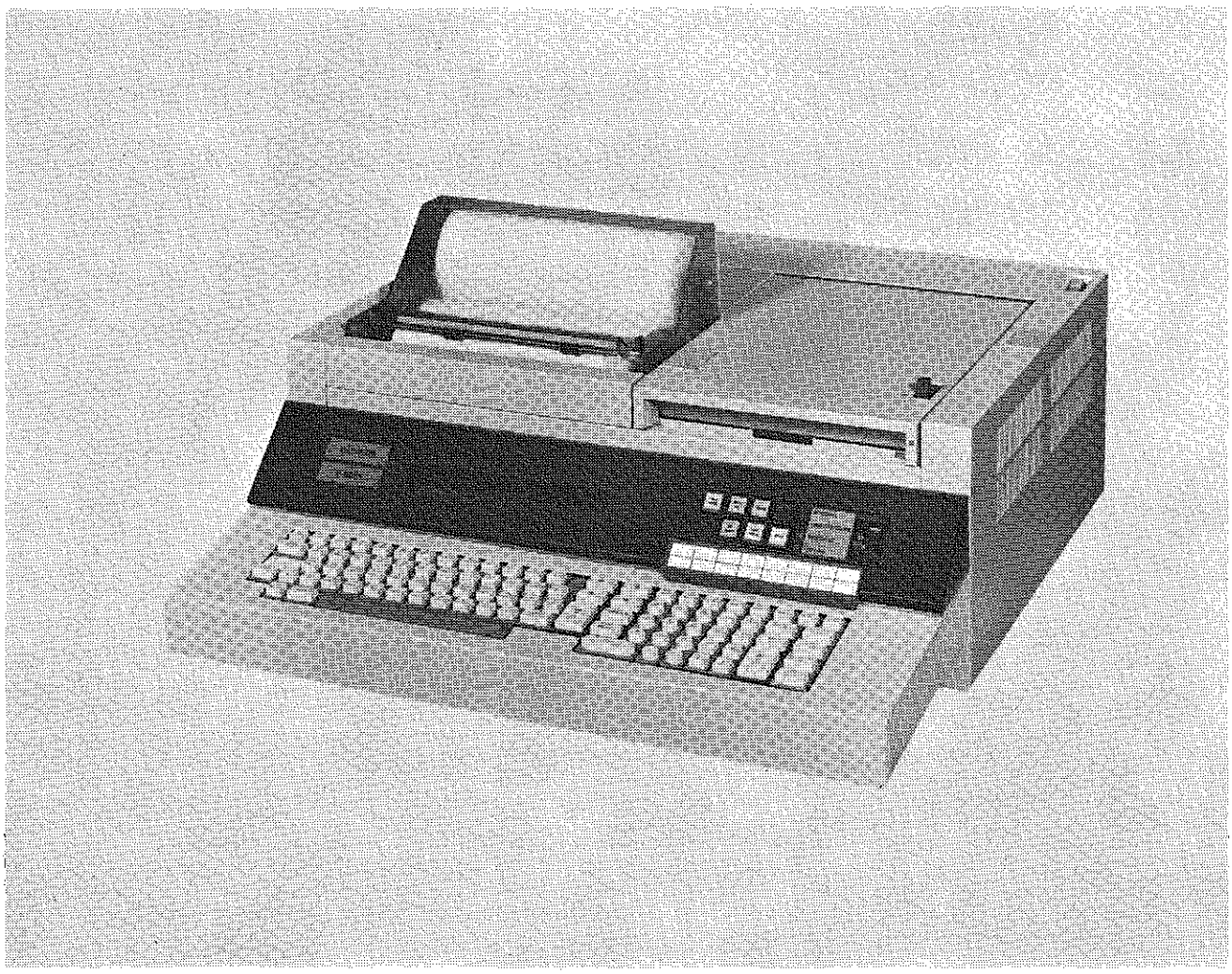


Figura 1-18 Sistema P6066 con la stampante integrata

La stampante integrata, vedi fig. 1-18, è di tipo termografico con caratteri di stampa (vedi set completo in Appendice E) composti su matrice di 5 x 7 punti.

Su ogni riga di stampa si possono stampare al massimo 80 caratteri con velocità di stampa di 120 caratteri/s . La stampante può tracciare grafici ed immagini comunque complesse, predefiniti con programmi scritti nel linguaggio BASIC, vedi manuale "P6066 Prestazioni grafiche - Manuale del programmatore".

Si può anche collegare un'unità Video Olivetti DSM 6680 che consente la visualizzazione di dati alfanumerici e la rappresentazione grafica di immagini su uno schermo video. Per informazioni vedere il manuale "P6066 Prestazioni grafiche - Manuale del programmatore".

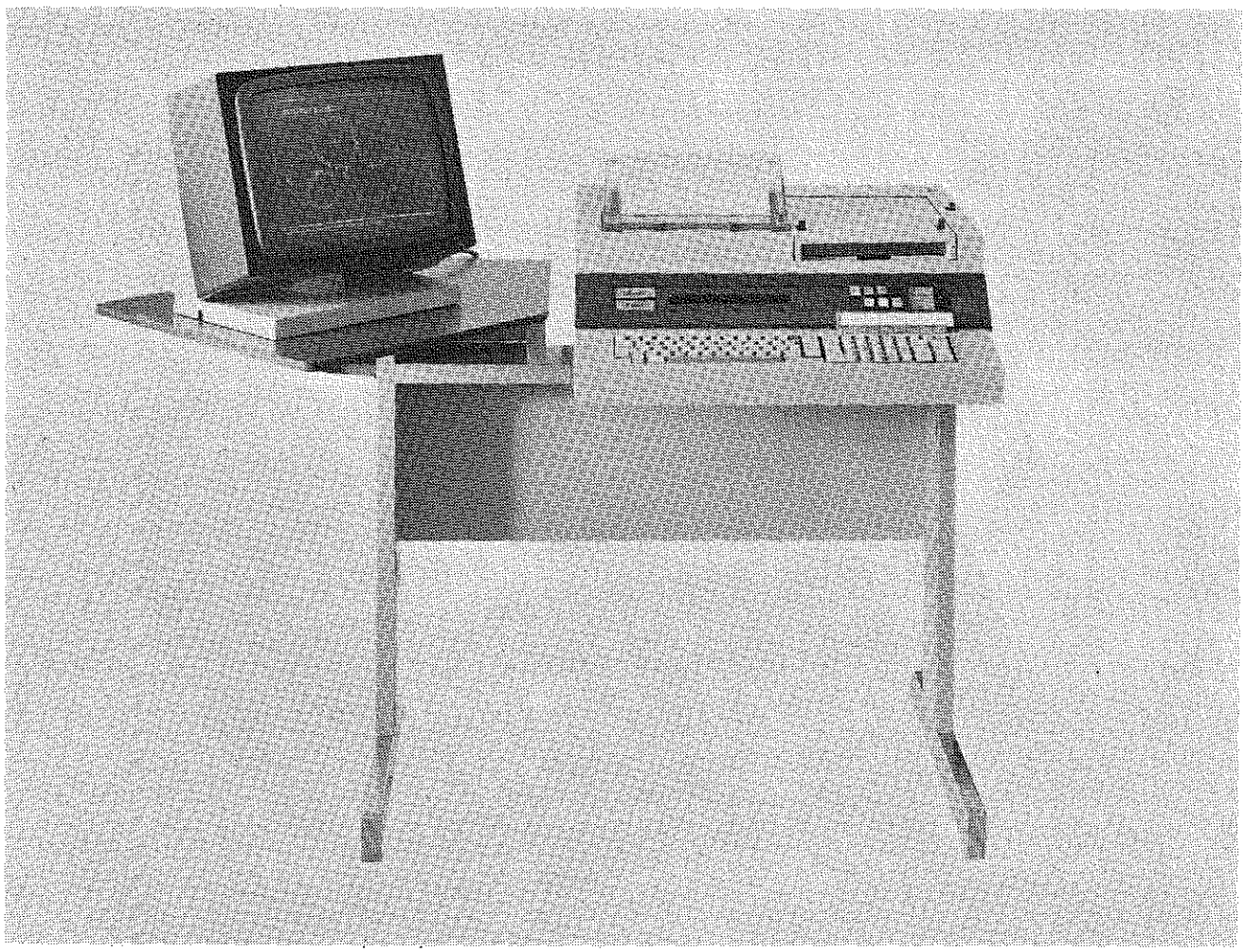


Figura 1-19 Il sistema P6066 con il video

Il secondo trascinatore per floppy disk consente l'utilizzazione di un floppy disk supplementare che incrementa di 204K byte (K=1024) lo spazio disponibile per la creazione di file e per la registrazione di programmi. In figura 1-20 è indicato un P6066 che utilizza due trascinatori floppy disk.

Figura 1-20 Unità floppy disk con 2 floppy disk

Per l'utilizzazione di periferiche esterne devono essere presenti nella cassettiera contenente i circuiti logici della macchina i governi di I/O relativi. Si tengano presenti i vincoli descritti nel manuale "P6066 Guida alla configurazione", codice 3977840 V.

Il governo IPSÒ 6600 permette di collegare al Sistema P6066 fino a quattro periferiche di input, output od input/output scelte fra i tipi seguenti:

- perforatore di nastro
- lettore di nastro
- lettore di schede
- lettore ottico
- unità a cassetta magnetica
- unità a nastro magnetico
- tracciatore di grafici
- stampante 90 caratteri/s
- stampante 100 caratteri/s
- stampante 175 caratteri/s
- stampante 200 caratteri/s
- stampante 300 caratteri/s

- governo linea asincrono

Il governo PIC 6626 permette il collegamento a periferiche compatibili con il sistema di interfaccia standard IEEE 488 - 1978.

Il governo SIC 6629 permette il collegamento a periferiche con interfaccia compatibile EIA RS232C (CCITT V24), inclusi i modem per la trasmissione di dati su linee, ed a periferiche compatibili 20 mA current loop. E' anche possibile il collegamento in time-sharing od in remote job entry a calcolatori remoti.

Per ulteriori informazioni sulle periferiche esterne si veda il manuale: "P6066 I/O con periferiche esterne - Manuale del programmatore", e i manuali della serie "P6066 I/O Driver Series" e "P6066 Data Communication Series".

2. COME USARE IL SISTEMA

In questo capitolo sono descritte le operazioni fondamentali che si possono eseguire con il sistema P6066. Nella parte iniziale si descrive come il sistema può essere acceso, spento e inizializzato. Poichè l'unità principale di comunicazione con il sistema è la tastiera si dedica ampio spazio alla descrizione del suo impiego. Quindi è definita brevemente la memoria utente, in quanto supporto fondamentale in cui risiede il programma utente durante l'elaborazione dei dati.
Dopo aver definito i diversi stati in cui può trovarsi il sistema specificando le operazioni che si possono eseguire in ogni stato, si descrivono le operazioni fondamentali di creazione, editing e registrazione di un programma utente o di un file testo.

Preparazione dei supporti

Prima di iniziare ad usare il sistema per eseguire un qualsiasi lavoro si devono predisporre i supporti di registrazione che saranno utilizzati durante l'impiego del sistema.

Per il montaggio dei supporti utilizzati dalle unità periferiche esterne (unità a cassette magnetiche, perforatori di nastro etc.) si vedano i manuali delle periferiche. Per quanto riguarda l'unità a disco si veda quanto detto nel capitolo 1 e, per maggiori particolari, il manuale dell'unità. Qui vengono descritte le operazioni di servizio che riguardano il cambio del rullo di carta della stampante integrata e l'inserimento dei floppy disk nei relativi trascinatori.

Cambio del rullo di carta

Per estrarre il rullo di carta presente sulla stampante integrata e sostituirlo con uno nuovo si osservino le figure 2-1 e 2-2 e si seguano le seguenti istruzioni.

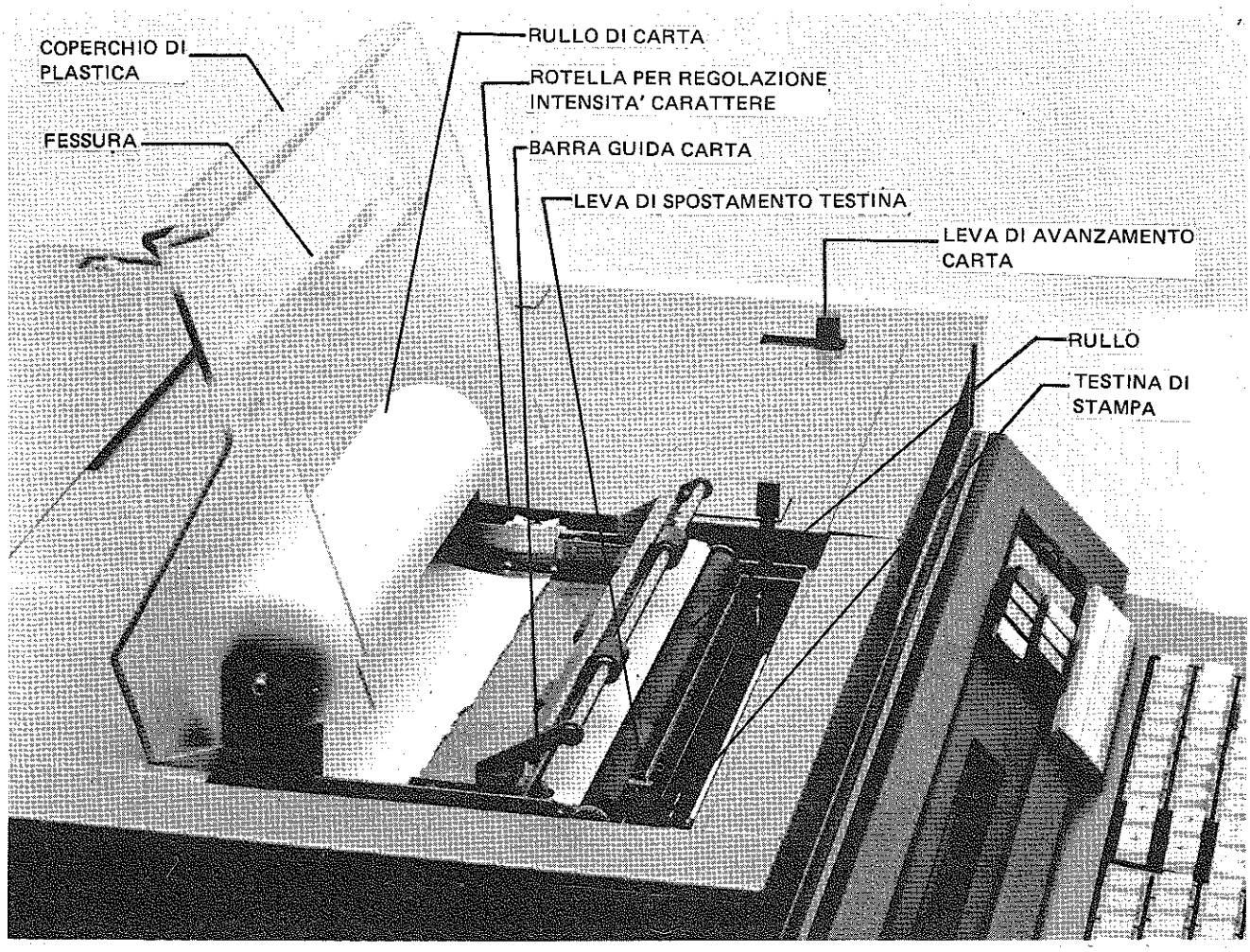


Figura 2-1 Parti componenti la stampante integrata

1. Sollevare il coperchio di plastica.

2. Allontanare la testina di stampa dalla carta tirando leggermente indietro la leva relativa.

3. Estrarre il rullo di carta dal suo alloggiamento ed estrarre dal rullo di carta il cilindro di metallo.

4. Introdurre il cilindro di metallo nel nuovo rullo di carta e poggiare il rullo nell'alloggiamento relativo della unità di stampa.

5. Far scorrere a mano la carta sotto il rullo di stampa e quindi verso l'alto mediante la leva di avanzamento (figura 2-2).

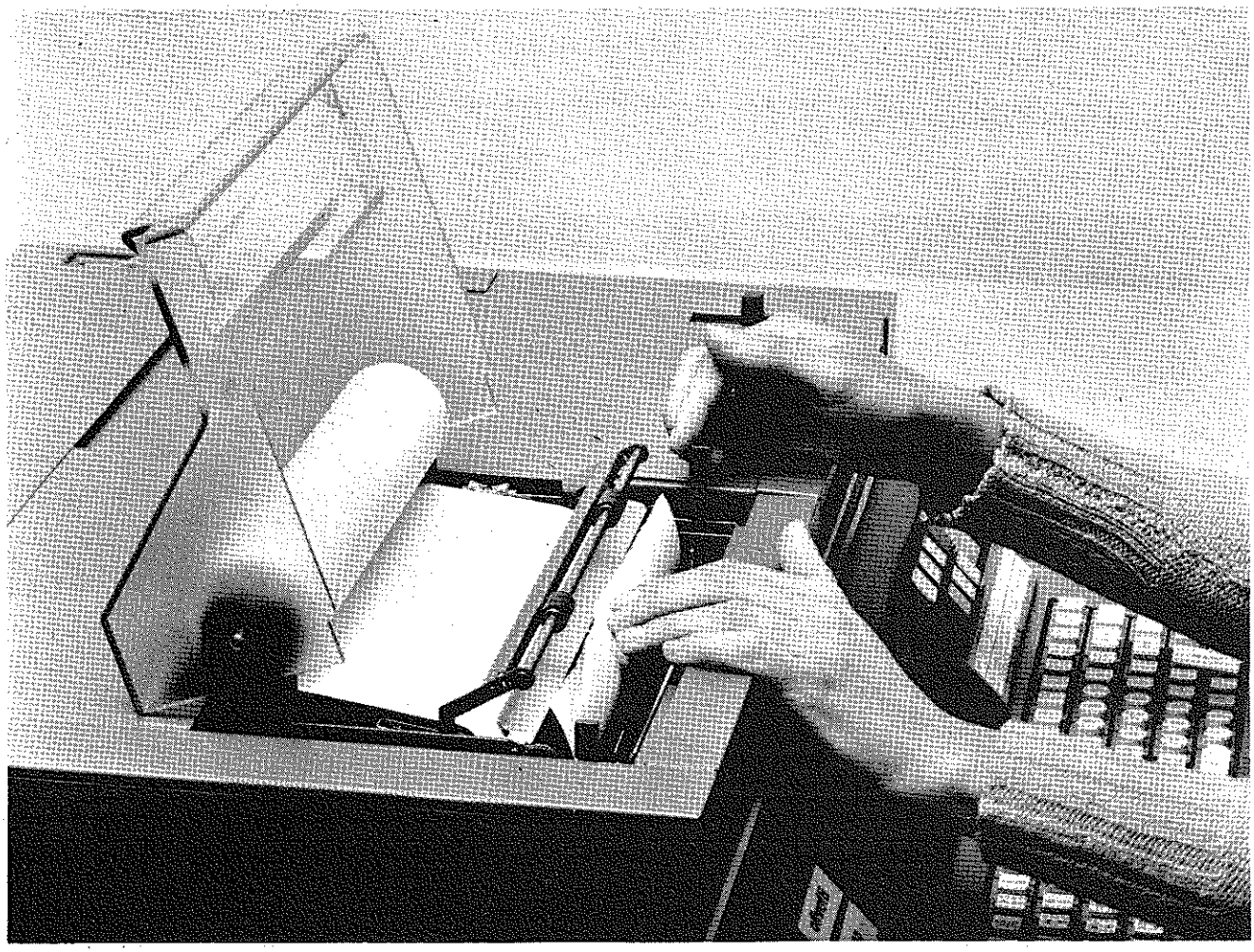


Figura 2-2 Avanzamento della carta nella stampante integrata

6. Quando la carta è avanzata di qualche centimetro, abbassare il ferma carta.

7. Avvicinare la testina di stampa alla carta spingendo leggermente in avanti la leva relativa.

8. Scegliere l'intensità di stampa desiderata ruotando la relativa rotella. L'intensità di stampa varia da 0 a 9; 0 produce il carattere più scuro e 9 il carattere più chiaro.

9. Abbassare il coperchio di plastica inserendo la carta nella feritoia relativa; tirare indietro la leva di avanzamento carta per controllare che lo scorrimento della carta attraverso la feritoria sia regolare.

Inserimento dei floppy disk

La procedura da seguire dipende dal tipo di unità floppy disk disponibile.

Se si ha una unità ad un solo trascinatore:

1. Accendere il sistema.

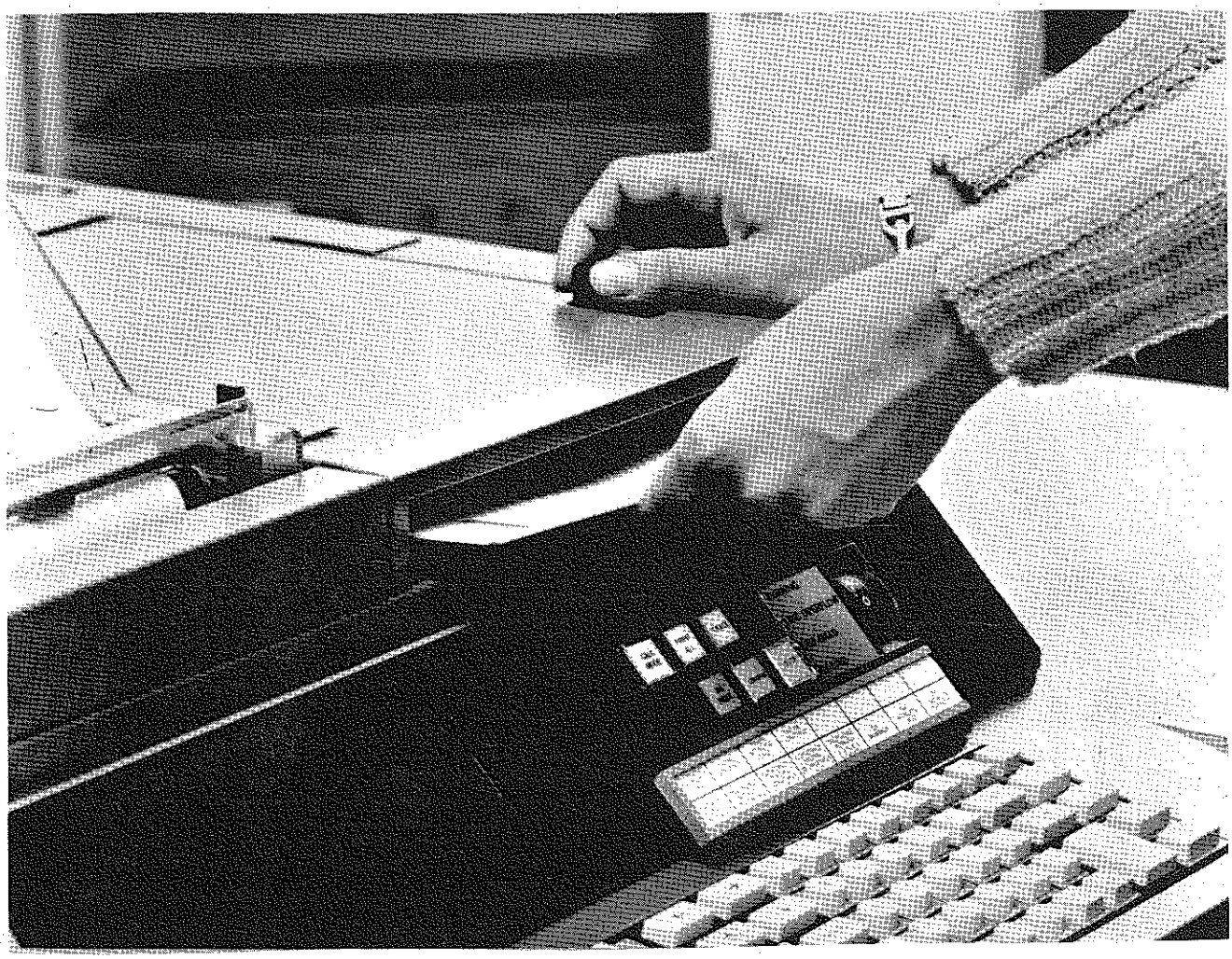


Figura 2-3 Inserimento del floppy disk in una unità monodisco

2. Inserire il floppy disk sistema o il floppy disk bootstrap nel trascinatore, con l'etichetta rivolta verso l'alto e verso l'operatore (come indicato in figura 2-3), finchè si sente un click.

3. Chiudere lo sportello tirando indietro dolcemente la relativa leva.
   Attenzione: Prima di chiudere lo sportello, ci si assicuri che il sistema sia acceso.

Se si ha un'unità con due trascinatori:

1. Accendere il sistema.

2. Sbloccare l'unità spingendo in avanti lu leva relativa. Sollevare l'unità. Aprire gli sportelli tirando indietro le relative leve.

3. Inserire un floppy disk nel trascinatore superiore con l'etichetta rivolta verso l'alto; inserire un altro floppy disk nel trascinatore inferiore con l'etichetta rivolta verso il basso, come indicato nella figura 2-4. (Il floppy disk sistema od il floppy disk bootstrap possono essere inseriti sia nel trascinatore superiore che in quello inferiore).

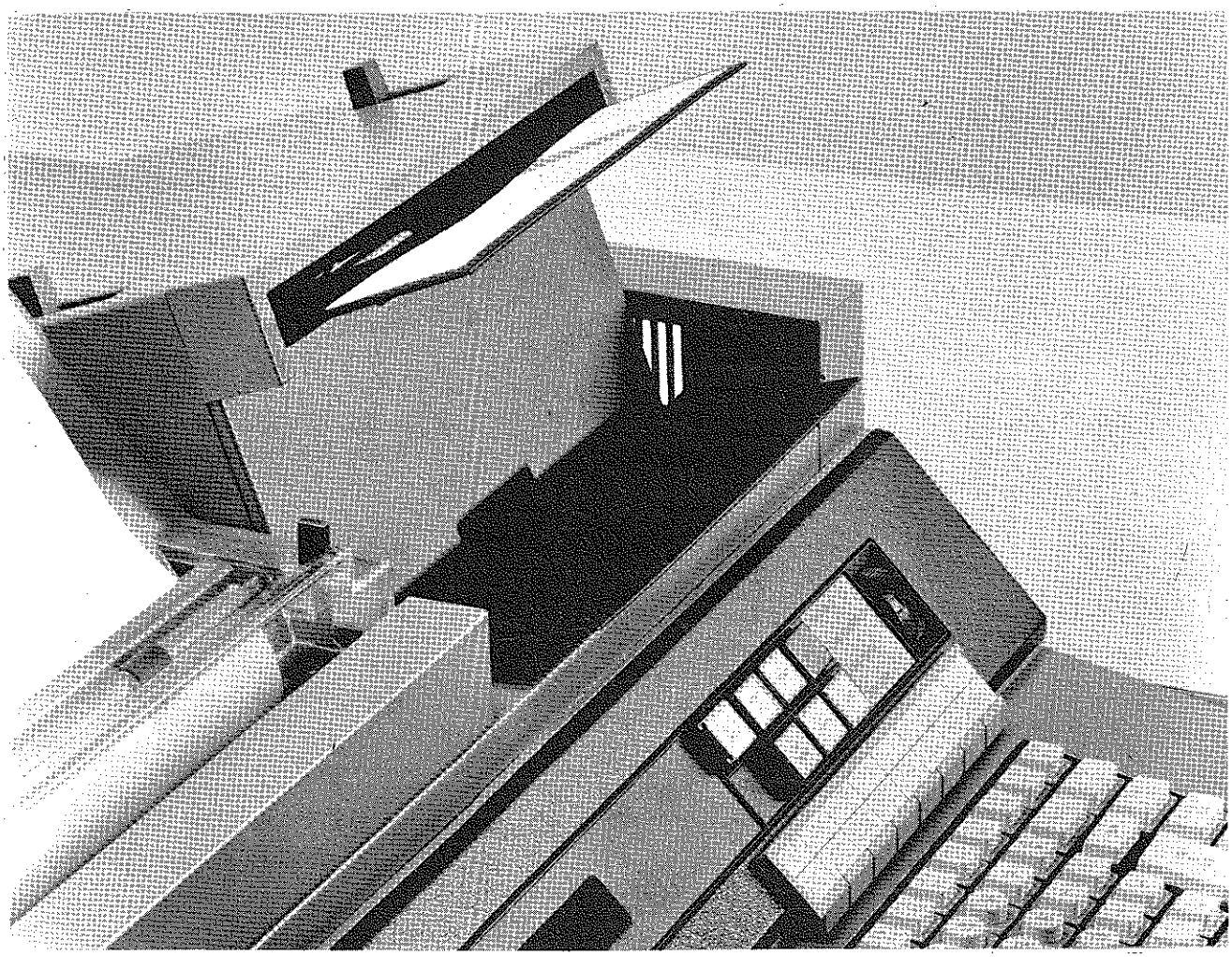


Figura 2-4 Inserimento del floppy disk in un'unità bidisco

4. Chiudere gli sportelli dell'unità tirando indietro dolcemente le relative leve.

   Attenzione: Prima di chiudere gli sportelli, ci si assicuri che il sistema sia acceso.

5. Abbassare l'unità.

Accensione

1. Premere l'interruttore ON/OFF dalla parte ON (ved. fig. 1-1). Se al P6066 è collegata una unità DCU si deve verificare che l'indicatore READY sia acceso. Se al P6066 è collegata un'unità HDU si dia prima tensione all'HDU.

2. Immediatamente dopo aver acceso il sistema tutte le luci di console si accendono ed il segnalatore acustico emette un suono.

3. Dopo qualche secondo le luci di console possono assumere diverse configurazioni. Le configurazioni assunte hanno i seguenti significati:

LUCI DI CONSOLE ACCESE	SIGNIFICATO
BREAK, RUNNING, NO PRINT, ON LINE	Manca il floppy disk nell'unità
TRACE, RUNNING, NO PRINT, ON LINE	Manca il floppy disk sistema od il floppy disk bootstrap nell'unità
NO PRINT, ON LINE, DEG GRAD	Floppy disk sistema o floppy disk bootstrap non corretto
BREAK, RUNNING, NO PRINT, DEG GRAD	Floppy disk sistema o floppy disk bootstrap non corretto
NO PRINT, RUNNING, ON LINE	Sportello unità floppy disk aperto

Nota: Per configurazioni diverse da quelle specificate sopra si contatti il più vicino servizio tecnico di assistenza Olivetti.

Inizializzazione

Dopo aver acceso il P6066 il sistema inizia una fase detta inizializzazione. Durante l'inizializzazione, la parte di sistema operativo necessaria per interpretare quanto viene introdotto da tastiera è trasferita dal disco sistema o floppy disk sistema in memoria principale. Durante questa fase la luce di console RUNNING lampeggia. Quando l'inizializzazione è completata, il messaggio READY è visualizzato sul display. La luce RUNNING smette di lampeggiare ma rimane accesa. Il sistema è ora pronto a ricevere comandi e istruzioni BASIC.

Nel caso in cui si utilizzi una configurazione di sistema con il sistema operativo residente su disco, al termine della inizializzazione, il floppy disk bootstrap può essere rimosso e sostituito con un floppy disk utente eseguendo il comando DCHANGE (vedi DCHANGE nel capitolo 3).

Introduzione da tastiera

Da tastiera si possono introdurre:

- comandi di sistema
- istruzioni BASIC
- stringhe di caratteri
- dati numerici
- linee di testo
- espressioni da eseguire immediatamente

I caratteri introdotti da tastiera sono memorizzati in un registro detto "buffer di tastiera" che ha una capacità di 80 caratteri. Il carattere introdotto è immediatamente visualizzato sul display alla destra della posizione indicata precedentemente dal pointer di display ed esso si sposta di una posizione verso destra.

Dal buffer di tastiera i caratteri sono trasferiti in memoria principale quando è premuto il tasto [END OF LINE] o [SUM], a prescindere dalla posizione del pointer sul display. I caratteri sul display sono cancellati ed il pointer si pone nella prima posizione del display.

I caratteri possono essere digitati in tastiera anche mentre il sistema sta eseguendo elaborazioni, operazioni di stampa, operazioni di I/O su disco o floppy disk, ma il comando END OF LINE o SUM è rifiutato dal sistema che emette una segnalazione acustica. Non appena la luce RUNNING è fissa i tasti [END OF LINE] e [SUM] sono abilitati.

Se si introducono, da tastiera, più di 80 caratteri l'ultimo carattere digitato è rifiutato, si ha una segnalazione acustica e si accende la luce di console LINE OVERFLOW.

Se si digitano contemporaneamente 2 caratteri la battuta è ignorata e si ha una segnalazione acustica.

In fig. 2-5 si vede che il display può visualizzare i caratteri introdotti da tastiera oppure messaggi di programma o di sistema. I messaggi di programma o di

sistema sono trasferiti dalla memoria principale ad un "buffer di display".

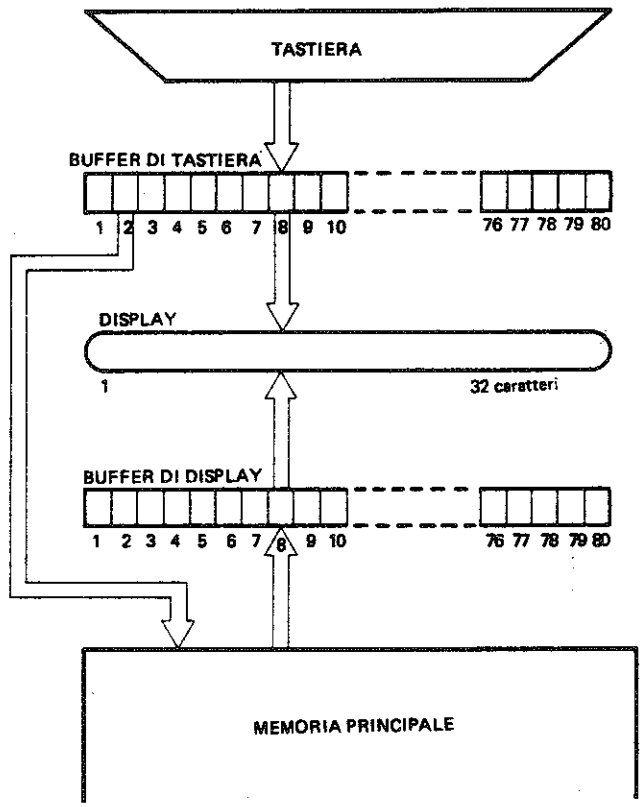


Figura 2-5 Visualizzazione sul display di caratteri introdotti da tastiera o generati da programma (o sistema)

Il display è collegato al buffer di display automaticamente dal sistema quando viene eseguita una istruzione DISP o il programma è in attesa di dati da tastiera (viene visualizzato ? oppure ??): istruzione INPUT, MAT INPUT o RKB. Non appena da tastiera viene digitato un carattere, il messaggio sul display è cancellato ed è visualizzato il contenuto del buffer di tastiera. Quando il display visualizza il contenuto del buffer di tastiera è possibile rivedere il contenuto del buffer di display premendo (CLEAR RECALL) e, viceversa, premendo lo stesso tasto si collega il buffer di tastiera con il display se quest'ultimo era collegato con il buffer di display.

Correzione delle introduzioni da tastiera

Prima di premere [END OF LINE] o [SUM] si possono cancellare, modificare o inserire caratteri nel buffer di tastiera.

1. Cancellazione:

   - per cancellare un carattere:

     . spostare il pointer di display nella posizione immediatamente a destra del carattere da cancellare utilizzando [←], [→], [SHIFT], [REPEAT].

     . premere [CHAR DELETE]. Il carattere è cancellato, il pointer ed i caratteri alla sua destra sono spostati di una posizione verso sinistra.

   - per cancellare tutti i caratteri presenti nel buffer di tastiera:

     . premere contemporaneamente i tasti [SHIFT] e [CLEAR RECALL] a prescindere dalla posizione del pointer. I caratteri nel buffer sono tutti cancellati ed il pointer si sposta nella prima posizione del display.

2. Modificazione:

   - per modificare un carattere:

     . cancellare il carattere da modificare. (Vedi cancellare un carattere)

     . digitare il carattere voluto. Il carattere digitato sostituisce nella posizione del buffer il carattere cancellato; il pointer è alla sua destra.

3. Inserimento:

   - per inserire un carattere:

     . spostare il pointer sul display nella posizione in cui si vuole inserire un nuovo carattere utilizzando [←], [→], [SHIFT], [REPEAT].

     . digitare il carattere da inserire nella stringa presente nel buffer di tastiera. Sul display il carattere viene visualizzato nella posizione

in cui è inserito ed il pointer è visualizzato alla sua destra.

Introduzione della data

Le operazioni sui file possono essere datate dall'utente mediante il comando DATE. Il formato generale del comando è: DATE date, dove "date" sono 6 caratteri che esprimono nell'ordine il giorno, il mese e l'anno (formato europeo) oppure il mese, il giorno e l'anno (formato americano). Il sistema associa la data specificata a qualunque file che viene registrato su disco o floppy disk. Così il comando DATE permette di ricordare quando un dato file è stato creato o modificato. (Tra i caratteri che compongono "date" non è ammesso lo spazio).

Working File

L'area di memoria principale occupata da un programma utente, un programma di utilità o da un file testo è definita working file. Nel working file può essere memorizzato un solo programma od un solo testo per volta. Il contenuto del working file viene cancellato quando si spegne il sistema. Dopo l'accensione del sistema, terminata la fase di inizializzazione, o dopo l'esecuzione dei comandi OPTIONS e CONFIGURE,il sistema inizializza il working file per l'eventuale introduzione di un programma da tastiera. Se in memoria principale esiste già un programma o file testo,per introdurre un nuovo programma da tastiera,si deve eseguire il comando NEW. Un file testo si può introdurre da tastiera solamente dopo che si è digitato il comando TEXT.

Opzioni del sistema operativo

Il sistema P6066 permette di eseguire programmi che elaborano matrici, che elaborano stringhe, che producono grafici, che impiegano periferiche seriali, che impiegano un video display, che impiegano una stampante IPSO in alternativa alla stampante integrata se l'utente lo specifica mediante un comando OPTIONS od un comando CONFIGURE. Ognuna delle prestazioni suddette richiede il caricamento in memoria utente di una specifica routine del sistema operativo.

Con il comando OPTIONS si possono caricare in memoria utente le seguenti routine del sistema operativo:

Nome	Funzione	Spazio di memoria utente richiesto (K=1024)
DEB	Permette l'esecuzione del debugging di un modulo assembler	3,5K byte
MAT	Permette di elaborare matrici con istruzioni MAT	1,5K byte
STR	Permette di eseguire le funzioni BLN$ e REP$	1K byte
PLO	Permette di tracciare grafici con la stampante integrata	2K byte
GDI	Permette di tracciare grafici sullo schermo del video Olivetti DSM 6680	2,5K byte
XNT	Permette la esecuzione di routine per la gestione di interruzioni da unità periferiche di I/O	1K byte
RS232	Permette l'impiego di periferiche seriali	1K byte

Per ulteriori informazioni si veda il comando OPTIONS nel capitolo 3.

Con il comando CONFIGURE si possono caricare in memoria utente le seguenti routine del sistema operativo:

Nome	Funzione	Spazio di memoria utente richiesto
EP	Permette l'impiego di una stampante IPSO in alternativa alla stampante integrata	0,5K byte

EVD	Permette l'impiego di un video display gestito in modo analogo alla stampante integrata	1K byte
EXD	Permette l'impiego di un video display Olivetti DSM 6680	1K byte

Per ulteriori informazioni si veda il comando CONFIGURE nel capitolo 3.

Ogni successiva inizializzazione del sistema, dopo la accensione, ricarica in memoria utente le routine del sistema operativo che erano state specificate con gli ultimi comandi OPTIONS e CONFIGURE eseguiti.

Stati del sistema

Per il sistema P6066 si hanno i seguenti "stati" o modi di funzionamento:

- stato comandi o di editing
- stato di esecuzione programma
- stato di debugging
- stato di esecuzione calcoli immediati.

Stato comandi

Nello stato comandi si può:

- digitare un comando di sistema (vedi capitolo 3)
- digitare un comando che richiama in memoria ed esegue un programma di utilità (vedi appendice A)
- premere il tasto di console CALC MODE (vedi capitolo 6)
- introdurre un programma BASIC

Il sistema è nello stato comandi:

- al termine della inizializzazione
- dopo la esecuzione di un comando di sistema

- dopo la esecuzione di un programma di utilità

- dopo la esecuzione di un programma utente

- se si richiede l'esecuzione di un programma che non può essere eseguito per insufficiente spazio in memoria principale

- se una operazione è stata interrotta dal comando di console BREAK

- se è stato premuto il tasto di console CALC MODE mentre il sistema era nello stato di esecuzione calcoli immediati

Quando il sistema è nello stato comandi la luce RUNNING è fissa.

Stato di esecuzione programma

Quando il sistema è nello stato di esecuzione programma esegue un programma utente, un programma di utilità o completa l'analisi sintattica di un programma. Il sistema è nello stato di esecuzione programma dopo la introduzione di:

- un comando RUN
- un comando PREPARE
- un comando EXEC

La luce RUNNING lampeggia quando il sistema è nello stato di esecuzione programma. Quando il sistema esegue un programma utente anche la luce del tasto CONTINUE è accesa.

Preesecuzione: Quando si introduce un programma BASIC da tastiera il sistema controlla la sintassi di ogni linea; gli errori di non coerenza sintattica tra diverse linee di programma non sono rilevati. Per esempio,se una istruzione GOTO fa riferimento ad un numero di linea inesistente,questo errore non è rilevato. Gli errori di questo tipo sono rilevati durante la fase di preesecuzione. Per preeseguire un programma si deve introdurre il comando PREPARE o RUN. Per tutti gli errori rilevati durante la preesecuzione sono stampati i relativi messaggi (vedi appendice D) ed il sistema commuta nello stato comandi. Dopo l'esecuzione del comando PREPARE, se non è stato rilevato alcun errore, il sistema è nello stato di debugging.

Esecuzione: Per eseguire un programma, si deve introdurre il comando RUN oppure premere il tasto di console CONTINUE dopo che è stata eseguita la preesecuzione comandata da un comando PREPARE. Se durante l'esecuzione è rilevato un errore recuperabile (ad esempio variabile non inizializzata) l'esecuzione del programma è interrotta ed è visualizzato il messaggio relativo (vedi appendice D). L'operatore può effettuare l'azione di recupero e quindi far riprendere l'esecuzione del programma premendo CONTINUE o STEP . Se durante l'esecuzione è rilevato un errore non recuperabile (vedi appendice D) l'esecuzione del programma è interrotta ed è visualizzato un messaggio. L'operatore può conoscere il valore assunto dalla variabile di programma introducendone il nome da tastiera; il valore viene immediatamente visualizzato sul display. L'operatore non può far riprendere l'esecuzione del programma ma deve premere il tasto di console BREAK per far terminare l'esecuzione.

I programmatori esperti che vogliono codificare delle loro routine di gestione degli errori possono usare le prestazioni disponibili con l'istruzione INTERRUPT ENABLE, che è descritta nel capitolo 6.

Interruzione della esecuzione di un programma: L'esecuzione di un programma è interrotta quando:

- è premuto il tasto STEP
- è eseguita l'istruzione STOP
- è rilevato un errore (recuperabile o no)
- è eseguita l'istruzione il cui numero di linea è specificato nel comando STOP introdotto durante lo stato di debugging

Ripresa dell'esecuzione: L'esecuzione di un programma prosegue quando:

- è premuto il tasto STEP nello stato di debugging
- è premuto il tasto CONTINUE
- è introdotto il comando START

Fine esecuzione: L'esecuzione di un programma termina quando:

- è eseguita l'istruzione END
- si preme il tasto BREAK

Quando è completata l'esecuzione di un programma il sistema è nello stato comandi.

Stato di debugging

E' lo stato del sistema che permette di verificare completamente la coerenza di un programma con l'algoritmo che esso traduce. Per una spiegazione dettagliata si veda il capitolo 7.

Stato di esecuzione calcoli immediati

E' lo stato del sistema che permette di eseguire immediatamente delle espressioni algebriche introdotte da tastiera dopo che si sia premuto il tasto END OF LINE o SUM . Per una spiegazione dettagliata si veda il - pitolo 6.

Creazione ed editing di un programma BASIC

Prima di vedere come si può introdurre ed editare da tastiera un programma BASIC, vediamone la struttura.

Struttura di un programma BASIC

Un programma BASIC è costituito da un insieme di linee (dette istruzioni); ognuna di esse contiene:

- un numero di linea
- una o più parole chiave BASIC
- uno o più operandi

Il numero di linea identifica ogni linea del programma.

Le parole chiave BASIC identificano la funzione della istruzione, ossia quale o quali azioni devono essere eseguite dal sistema.

Gli operandi specificano su quale variabile o espressione deve essere compiuta l'azione identificata dalla parola chiave BASIC o con quali condizioni o modalità.

L'ultima linea di un programma BASIC deve essere una istruzione END. Ecco un esempio di programma BASIC:

```
0010 REM AREA DI UN RETTANGOLO
0020 DISP "LUNGHEZZA E LARGHEZZA";
0030 INPUT L,W
0040 IF L<=0 THEN 20
0060 IF W<=0 THEN 20
0070 LET A=L*W
0080 PRINT "AREA=";A,L,W
0090 DISP "ANCORA:SI=1,NO=0";
0100 INPUT X
0110 IF X=1 THEN 20
0120 END
```

Introduzione ed editing di un programma

L'introduzione da tastiera di un programma avviene linea per linea. Ogni linea viene analizzata dal sistema dopo la pressione di [END OF LINE] . Se la linea introdotta è coerente con la sintassi (vedi capitolo 5), il sistema la traduce in codice oggetto che viene memorizzato nel working file. Se la linea introdotta non è coerente con la sintassi del linguaggio, il sistema non la accetta e sul display appare un messaggio di errore (vedi appendice D). La linea introdotta rimane memorizzata nel buffer di tastiera e non viene trasferita nel working file. Premendo il tasto [CLEAR RECALL] si può richiamare sul display la linea introdotta per poter effettuare le necessarie correzioni. La linea riappare sul display con il pointer vicino al carattere o ai caratteri errati. Dopo aver corretto l'errore, si può restituire al sistema la linea corretta, a prescindere dalla posizione del pointer premendo: [END OF LINE] .

I seguenti tasti e comandi possono essere usati per modificare un programma od un file testo:

- tasti: [←], [→], [↑], [↓], [SHIFT], [CHAR DELETE], [CLEAR RECALL], [END OF LINE]
- comandi: DELETELINE, FETCH, RESEQUENCE

Per meglio comprendere le tecniche di editing supponiamo di introdurre da tastiera il seguente programma BASIC dopo aver premuto il tasto di console [PRINT ALL] per attivare la relativa funzione. Nell'esempio riportato i punti significativi sono indicati con numeri riportati nel margine sinistro.

```
1 NEW
2 AUTO#
3 10 DIZP"INTRODUCI I COEFFICIENTI";
  ERROR  102
4 10 DISP"INTRODUCI I COEFFICIENTI";
  20 INPUTA,B,C
  30 I1=0
5 AUTO#
  40 I2=0
  50 IFA=0THEN310
  60 R=-B/(2*A)
  70 D=R*R-C/A
  80 X1=X2=R
  90 IFD>0THEN230
  100 IFD<0THEN150
  110 PRINTTAB(35);"*RADICI EGUALI*"
  120 DISP
  130 PRINTTAB(40);"X1=";X1,"X2=";X2
  140 GOTO400
  150 D=SQR(ABS(D))
  160 I1=-D
  170 I2=B
  180 PRINTTAB(35);"*RADICI COMPLESSE*"
  190 PRINT
  200 PRINT
  210 PRINT"X1-iI1=";X1;"-i";D,"X2+iI2=";X2;"+i";D
  220 GOTO400
  230 D=SQR(D)
  240 X1=X1-D
  250 X2=X2+D
  260 PRINTTAB(75);"*DUE RADICI*"
  270 DISP
  280 PRINT
  290 PRINTTAB(30);"X1=";X1,"X2=";X2
  300 GOTO400
  310 IFB=0THEN360
  320 X1=-C/B
  330 PRINTTAB(35);"*UNA RADICE*"
  340 PRINTTAB(30);"X1=";X1
  350 GOTO400
  360 IFC=0THEN390
  370 PRINT"*SOLUZIONE IMPOSSIBILE*"
  380 GOTO400
  390 PRINTTAB(35);"*SOLUZIONI INDETERMINATE*"
  400 DISP"UN'ALTRA EQUAZIONE"
  410 INPUTA$
  420 IFA$="SI"THEN10
6 430 END
```

1. Si introduce il comando NEW perchè in memoria principale c'è già un programma.

2. Si digita il comando AUTO#. AUTO# genera la numerazione automatica delle linee di programma con incremento di 10 -- partendo da 10 --. Sul display appare il numero 10.

3. Il sistema visualizza un messaggio di errore sintattico (vedi appendice D) che indica l'introduzione di una parola chiave non corretta: DIZP. Premendo il tasto [CLEAR RECALL] sul display appare:

   10 DI$_{\circ}$ZP "INTRODUCI I COEFFICIENTI"

   Premere [S]. Sul display appare:

   10 DIS$_{\circ}$ZP "INTRODUCI I COEFFICIENTI"

   Premere [→]. Sul display appare:

   10 DISZ$_{\circ}$P "INTRODUCI I COEFFICIENTI"

   Premere [CHAR DELETE]. Sul display appare:

   10 DIS$_{\circ}$P "INTRODUCI I COEFFICIENTI"

   Premere [END OF LINE].

4. La linea è accettata e trasferita nel working file. Si introduce la seconda istruzione e quindi le successive.

5. Come linea 40 l'operatore ha introdotto A2=5 ma non ha ancora premuto [END OF LINE]. Premere [CLEAR RECALL] con [SHIFT]. La linea nel buffer di tastiera è cancellata ed il pointer è in prima posizione. E' interrotta la numerazione automatica dei numeri di linea. Premere [AUTO#] e [END OF LINE]. Sul display appare il numero 40; è ripristinata la numerazione automatica dei numeri di linea. Si prema [1] [2] [=] [0].
   Si introducono le linee successive.

6. E' introdotta l'istruzione END. Premere [CLEAR RECALL] con [SHIFT]: la numerazione automatica è interrotta. Il sistema è nello stato comandi: si può introdurre qualunque comando. Il programma è presente nel formato eseguibile nel working file: può essere

eseguito o modificato.

Premere (1) (2) (5) [SHIFT] [PRINT] [END OF LINE] .
La linea introdotta è inserita nel working file tra la linea 120 e la linea 130.

Premere [R] [E] [S] [E] [Q] [U] [E] [N] [C] [E] [END OF LINE]
Le linee di programma sono rinumerate iniziando da 10 con passo 10. Tutti i riferimenti ad istruzioni di programma contenuti in istruzioni di salto sono rinumerati automaticamente come si vede nel listing seguente. Premere [LIST] [END OF LINE] .

```
0010 DISP "INTRODUCI I COEFFICIENTI";
0020 INPUT A,B,C
0030 LET I1=0
0040 LET I2=0
0050 IF A=0 THEN 320
0060 LET R=-B/(2*A)
0070 LET D=R*R-C/A
0080 LET X1=X2=R
0090 IF D>0 THEN 240
0100 IF D<0 THEN 160
0110 PRINT TAB(35);"*RADICI EGUALI*"
0120 DISP
0130 PRINT
0140 PRINT TAB(40);"X1=";X1,"X2=";X2
0150 GOTO 410
0160 LET D=SQR(ABS(D))
0170 LET I1=-D
0180 LET I2=B
0190 PRINT TAB(35);"*RADICI COMPLESSE*"
0200 PRINT
0210 PRINT
0220 PRINT "X1-iI1=";X1;"-i";D,"X2+iI2=";X2;"+i";D
0230 GOTO 410
0240 LET D=SQR(D)
0250 LET X1=X1-D
0260 LET X2=X2+D
0270 PRINT TAB(75);"*DUE RADICI*"
0280 DISP
0290 PRINT
0300 PRINT TAB(30);"X1=";X1,"X2=";X2
0310 GOTO 410
0320 IF B=0 THEN 370
0330 LET X1=-C/B
0340 PRINT TAB(35);"*UNA RADICE*"
0350 PRINT TAB(30);"X1=";X1
0360 GOTO 410
0370 IF C=0 THEN 400
0380 PRINT "*SOLUZIONE IMPOSSIBILE*"
0390 GOTO 410
0400 PRINT TAB(35);"*SOLUZIONI INDETERMINATE*"
0410 DISP "UN'ALTRA EQUAZIONE"
0420 INPUT A$
0430 IF A$="SI" THEN 10
0440 END

END OF LISTING
```

Si osservi che anche se le istruzioni di assegnazione sono introdotte senza il verbo BASIC LET, il sistema lo stampa nel listing. Il sistema infatti stampa il listing del programma eseguendo un editing tale da

rendere facilmente leggibili le istruzioni. Lo stesso editing viene effettuato per le linee di programma visualizzate sul display con il comando FETCH od i tasti [↑] e [↓].

Premere [SHIFT] [FETCH] (1) (2) [0] [END OF LINE]
Sul display appare: 120 DISP

Premere [SHIFT] [→] ; premere quattro volte [CHAR DELETE]
Sul display si ha: 120

Premere [SHIFT] [PRINT]
Sul display appare: 120 PRINT

Premere [END OF LINE]
Nel programma la linea 120 precedente è sostituita da quella appena introdotta.

Premere [SHIFT] [DELETE LINE] (2) (8) [0] [END OF LINE]
La linea 280 è cancellata.

Premere [LIST] [END OF LINE]

Viene stampato il nuovo listing del programma:

```
LIST
FILE

0010 DISP "INTRODUCI I COEFFICIENTI";
0020 INPUT A,B,C
0030 LET I1=0
0040 LET I2=0
0050 IF A=0 THEN 320
0060 LET R=-B/(2*A)
0070 LET D=R*R-C/A
0080 LET X1=X2=R
0090 IF D>0 THEN 240
0100 IF D<0 THEN 160
0110 PRINT TAB(35);"*RADICI EGUALI*"
0120 PRINT
0130 PRINT
0140 PRINT TAB(40);"X1=";X1,"X2=";X2
0150 GOTO 410
0160 LET D=SQR(ABS(D))
0170 LET I1=-D
0180 LET I2=B
0190 PRINT TAB(35);"*RADICI COMPLESSE*"
0200 PRINT
0210 PRINT
0220 PRINT "X1-iI1=";X1;"-i";D,"X2+iI2=";X2;"+i";D
0230 GOTO 410
0240 LET D=SQR(D)
0250 LET X1=X1-D
0260 LET X2=X2+D
0270 PRINT TAB(75);"*DUE RADICI*"
0290 PRINT
0300 PRINT TAB(30);"X1=";X1,"X2=";X2
```

```
0310 GOTO 410
0320 IF B=0 THEN 370
0330 LET X1=-C/B
0340 PRINT TAB(35);"*UNA RADICE*"
0350 PRINT TAB(30);"X1=";X1
0360 GOTO 410
0370 IF C=0 THEN 400
0380 PRINT "*SOLUZIONE IMPOSSIBILE*"
0390 GOTO 410
0400 PRINT TAB(35);"*SOLUZIONI INDETERMINATE*"
0410 DISP "UN'ALTRA EQUAZIONE"
0420 INPUT A$
0430 IF A$="SI" THEN 10
0440 END

END OF LISTING
```

Il programma può essere registrato su disco o floppy disk utilizzando il comando SAVE (vedi capitolo 3).

Creazione ed editing di un file testo

Struttura di un file testo

Un file testo è composto da un insieme di linee, ognuna delle quali contiene un numero di linea seguito da caratteri scelti nel set P6066 (vedi appendice E). Ogni linea può contenere al massimo 80 caratteri (compreso il numero di linea). Ecco un esempio di file testo:

```
LIST
FILE    +TESTO

0010 QUESTO E' UN ESEMPIO DI FILE TESTO.
0020 Il file testo e' composto da un insieme di linee
0030 che iniziano con un numero di linea.
0040
0050 Il numero di linea e' molto importante perche' permette di richiamare nel
0060 buffer di tastiera le linee di testo che si vogliono
0070 modificare.
0080
0090 Per introdurre da tastiera un file testo si deve prima digitare
0100 il comando TEXT

END OF LISTING
```

I file testo sono utili in applicazioni quali -- per citarne alcune -- analisi linguistiche, la preparazione di documenti e la generazione di archivi commerciali ed amministrativi. Inoltre i file testo possono essere usati come strumenti di programmazione per:

1. Creare un file dati editabile successivamente utilizzato da un programma BASIC.

2. Creare programmi, sottoprogrammi o definizioni di funzione in linguaggio BASIC, da registrare su floppy disk o disco nel "formato sorgente", per essere successivamente tradotti nel formato eseguibile mediante il comando COMPILE o inseriti in programmi: comando LINK.

3. Creare programmi con un qualsiasi linguaggio di programmazione per la soluzione di particolari applicazioni. Così un file testo può essere composto da frasi come:

   P1 = L1, L2
   C3 = X30, Y50, R15.3

   che definiscono particolari situazioni geometriche utilizzando il linguaggio GTL (Geometrical and Technological Language) nell'ambito del controllo numerico. Considerando che una volta introdotte le linee del testo, sono possibili tutte le operazioni di editing proprie del sistema, si possono avere dei programmi che possono essere successivamente verificati.

Per una completa descrizione dell'impiego dei file testo si veda la pubblicazione P6066 - Gestione dei file testo, codice 3977870 Y.

Introduzione ed editing di un testo

L'introduzione da tastiera di un file testo avviene linea per linea. Il sistema verifica che ogni linea sia preceduta da un numero di linea ed in caso affermativo la trasferisce dal buffer di tastiera nel working file. Se la linea non è preceduta da un numero di linea viene fornita una segnalazione di errore sul display ed emessa una segnalazione acustica. Il trasferimento della linea nel working file non avviene. Introducendo il numero di linea all'inizio della linea, una successiva pressione di [END OF LINE] trasferisce la linea nel working file. Il numero di linee che si possono introdurre dipende dalla capacità della memoria utente disponibile.

Quando le linee del file testo sono presenti nel working file si possono ulteriormente editare utilizzando i seguenti tasti e comandi:

- tasti: ← → ↑ ↓ SHIFT CHAR DELETE CLEAR RECALL END OF LINE

- comandi: DELETE LINE, FETCH, RESEQUENCE

Per l'impiego degli strumenti di editing si veda l'esempio di editing del progrumma precedente. Quando il file testo è pronto lo si può registrare permanentemente sul disco o floppy disk utilizzando il comando SAVE (vedi capitolo 3).

Spegnimento

1. Premere il tasto di console BREAK prima di spegnere il sistema se è in corso di esecuzione un programma che elabora file dati o testo (vedi capitolo 3) e quindi attendere che la luce incorporata nel tasto sia spenta.

2. Per spegnere il P6066 premere l'interruttore ON/OFF dalla parte OFF. Se al P6066 è collegata una unità DCU, si deve prima premere il tasto LOAD/RUN dalla parte LOAD ed attendere che l'indicatore LOAD sia acceso.

3. Quando si spegne il P6066 non è necessario spegnere l'unità HDU, anzi è bene non accendere e spegnere più volte nell'arco della giornata tale unità.

3. COMANDI DI SISTEMA

La comunicazione fra l'utente ed il sistema è realizzata mediante un linguaggio costituito da comandi di sistema che permettono la creazione ed esecuzione di un programma BASIC in modo semplice ed immediato. Con i comandi disponibili si possono -- tra le altre cose:

- creare, modificare, eseguire e registrare programmi
- eseguire procedure
- creare e modificare file dati e file testo
- gestire librerie e sottolibrerie

Come usare i comandi di sistema è l'argomento di questo capitolo. Prima di entrare nei dettagli sui singoli comandi, tuttavia, sarà utile familiarizzare con i concetti ed i termini associati al linguaggio dei comandi. Questo è lo scopo dei paragrafi che seguono.

I dischi

Vi sono due tipi di dischi: sistema ed utente.

Disco sistema: Per disco sistema s'intende un disco che contiene il sistema operativo e risiede nella sezione dell'unità a disco specificata durante la generazione del sistema operativo (vedi l'Appendice B).

Il sistema operativo è composto da tre diverse librerie: la libreria di firmware residente (di nome KØEØØ2), la libreria di firmware opzionale (di nome KØEØØ3) e la libreria di software base (di nome KØEØØ1). Lo spazio rimanente sul disco sistema è disponibile all'utente per la creazione di 35 librerie che sono chiamate librerie utente.

Disco utente: Per disco utente s'intende qualunque disco che risiede in una sezione dell'unità a disco non specificata per contenere il disco sistema durante la generazione del sistema operativo. Un disco che risiede in tale sezione è trattato come disco utente anche se contiene un sistema operativo. Anche su un disco utente si possono registrare fino ad un massimo di 35 librerie utente.

I floppy disk

Vi sono tre tipi di floppy disk: bootstrap, sistema ed utente.

Floppy disk bootstrap: Un floppy disk bootstrap è usato solamente con sistemi che hanno il sistema operativo residente su disco. Esso contiene programmi necessari per eseguire l'inizializzazione del sistema. Su di esso non si possono registrare programmi o dati.

Floppy disk sistema: Per floppy disk sistema s'intende un floppy disk che contiene il sistema operativo. Il sistema operativo è composto da tre diverse librerie: la libreria di firmware residente (di nome KØEØØ2), la libreria di firmware opzionale (di nome KØEØØ3) e la libreria di software base (di nome KØEØØ1). Lo spazio rimanente è disponibile per la creazione di una libreria di utente in cui si possono registrare programmi e dati.

Floppy disk utente: Per floppy disk utente s'intende qualsiasi floppy disk su cui non vi sia registrato software di sistema. L'intero spazio è quindi disponibile per la creazione di una libreria di utente in cui possono essere registrati programmi e dati.

Nota: Se, al momento dell'accensione della macchina, vi sono due floppy disk sistema nell'unità, il disco nel trascinatore inferiore è considerato il floppy disk sistema.

Nomi per le unità a disco e floppy disk

Nei comandi di sistema un disco o floppy disk è riferito mediante il nome dell'unità in cui risiede. I nomi a tali unità sono assegnati durante la generazione del sistema operativo (vedi Appendice B). I nomi suggeriti come standard sono:

FDU1 per il trascinatore superiore dell'unità floppy disk

FDU2 per il trascinatore inferiore dell'unità floppy disk

LD per la sezione dell'unità DCU collegata direttamente al P6066 che contiene il disco fisso

UD per la sezione dell'unità DCU collegata direttamente al P6066 che contiene il disco rimovibile

SLD per la sezione dell'unità DCU collegata al P6066 attraverso un altro DCU che contiene il disco fisso

SUD per la sezione dell'unità DCU collegata al P6066 attraverso un altro DCU che contiene il disco rimovibile

HD per l'unità HDU collegata direttamente al P6066

SHD per l'unità HDU collegata al P6066 attraverso un altro HDU

Si noti che nel caso di unità a floppy disk con un solo trascinatore il nome dell'unità è FDU1.

Eseguendo il comando ENVIRONMENT vengono stampati i nomi delle unità a disco e floppy disk che sono collegate al sistema.

Organizzazione delle informazioni su disco o floppy disk

Le informazioni registrate su disco o floppy disk sono organizzate nella seguente struttura gerarchica:

librerie
sottolibrerie
file

Librerie

La libreria è la struttura informativa più complessa che i comandi di sistema possono gestire. Essa si compone di sottolibrerie a loro volta costiuite da file.

In una libreria possono essere distinte tre sottolibrerie che si differenziano per il livello di protezione ad esse associato.

Per poter eseguire le operazioni sui file di una libreria specificate da comandi di sistema, da istruzioni di un programma BASIC od eseguire i programmi di utilità FLCOPY e FLPRINT, la libreria di cui fanno parte deve essere stata "aperta", cioè dichiarata accessibile per le successive operazioni. Per aprire una libreria si deve eseguire il comando LBOPEN. Nei sistemi che hanno il sistema operativo residente su disco vi possono essere fino a sei librerie contemporaneamente aperte. Ogni volta che si spegne il sistema, o si eseguono i comandi CONFIGURE ed OPTION, tutte le librerie vengono chiuse. Quando si riaccende

il sistema, o dopo che è terminata l'esecuzione dei comandi CONFIGURE od OPTION, sono riaperte automaticamente le librerie specificate nell'ultimo comando LBSTORE eseguito.

Il nome delle librerie aperte, e l'ordine con cui sono state aperte, si può conoscere in ogni istante eseguendo il comando ENVIRONMENT od il comando LVTOC con * come operando. Infatti tali comandi stampano i nomi delle librerie aperte nello stesso ordine in cui sono state aperte. L'ordine con cui le librerie sono aperte è importante perchè la ricerca dei file programma, testo e dati contenuti nelle sottolibrerie avviene secondo tale ordine (vedi in seguito la definizione dei suddetti file).

Per determinare il nome delle librerie residenti su un disco o floppy disk si deve eseguire un comando LVTOC specificando il nome dell'unità in cui esso risiede.

Nomi e parole d'ordine

Quando una libreria è creata (vedi Appendice A il programma di utilità LBCREATE) ad essa è assegnato un nome ed, opzionalmente, una parola d'ordine. Il nome è costituito da una stringa di al massimo sei caratteri alfanumerici, dei quali il primo deve essere alfabetico: la stringa è specificata nella sintassi dei comandi per riferirsi alla libreria su cui il comando o programma di utilità deve agire.

La parola d'ordine è costituita da una stringa di al massimo sei caratteri del set P6066 (vedi Appendice E) diversi dallo spazio e dalla virgola. Alcune operazioni sulla libreria potranno essere eseguite solamente se sarà specificata la parola d'ordine ad essa associata (vedi nel seguito i comandi di sistema ed, in Appendice A, i programmi di utilità). La parola d'ordine può essere modificata od assegnata ad una libreria creata senza parola d'ordine, eseguendo il programma di utilità LBRENAME.

Una parola d'ordine di sistema può essere assegnata in fase di generazione di sistema nel caso di sistemi che hanno il sistema operativo residente su disco. La parola d'ordine di sistema può essere usata al posto di una parola d'ordine di qualsiasi libreria.

Sottolibrerie

Una sottolibreria è composta da un insieme di file. Le sottolibrerie che possono essere presenti in una libreria sono:

sottolibreria package
sottolibreria comune
sottolibreria utente

Sottolibreria package: Una sottolibreria package è composta da file i cui nomi iniziano con il carattere *. Il suo nome deriva dal fatto che i programmi dei Package di Software Applicativo distribuiti dalla Olivetti sono registrati in questo tipo di sottolibrerie.

Una sottolibreria package può essere protetta, eseguendo il programma di utilità LBPROTECT, dall'azione dei seguenti comandi:

CREATE
MODIFY (modifica del nome di un file)
PURGE
SAVE
TRANSCODE (con l'operando D)

e dei programmi di utilità FLCOPY e LIBCOPY.

Sottolibreria comune: Una sottolibreria comune è composta da file i cui nomi iniziano con il carattere +. Essa può essere protetta, eseguendo il programma di utilità LBPROTECT, dalla azione dei seguenti comandi:

MODIFY (modifica del nome di un file)
PURGE

Sottolibreria utente: Una sottolibreria utente è composta da file i cui nomi iniziano con una lettera maiuscola dell'alfabeto inglese. Una sottolibreria utente non può essere protetta, quindi in essa si possono sempre cancellare o aggiungere file e modificarne i nomi.

Nota: Si osservi che la protezione di una sottolibreria è un'operazione distinta dalla protezione delle informazioni contenute nei singoli file. I singoli file possono essere ulteriormente "assicurati" contro l'azione di alcuni comandi o programmi di utilità eseguendo il comando SECURE.

File

In ogni sottolibreria vi possono essere tre diversi tipi di file: file programma, file testo e file dati.

File programmi: Come già definito nel capitolo 2, un file programma è composto da una serie di istruzioni BASIC, ognuna preceduta da un numero di linea. L'istruzione finale di un programma deve essere l'istruzione END. (Il termine "file programma" può essere usato al posto del termine "programma").

File testo: Come già definito nel capitolo 2, un file testo è un insieme di linee numerate e memorizzate in memoria in formato sorgente -- il formato con cui sono introdotte. Ogni linea può contenere un qualsiasi carattere del set P6066.

File dati: Un file dati è composto da un insieme di dati numerici o stringa, di solito usati come input di un programma BASIC. Un file dati può essere creato in due modi: (1) come output di un programma BASIC, (2) usando, in combinazione, i comandi TEXT e TRANSCODE. In funzione del modo con cui i dati sono indirizzati, il file può essere sia sequenziale che ad accesso diretto. (Per ulteriori informazioni sui file dati, si veda il capitolo 4.)

Protezione dei programmi e file dati: Il P6066 permette di assicurare contro un accesso non autorizzato i file programma e file dati. Utilizzando questa prestazione si possono creare programmi che possono essere solamente eseguiti o copiati -- non letti, listati o modificati in alcun modo. Tale protezione è ottenuta per mezzo del comando SECURE.

Nomi di file: Ogni file registrato in una libreria è identificato da un nome. Il nome del file permette la ricerca di esso e la sua protezione da un accesso non autorizzato. Lo stesso nome può essere assegnato a due diversi file purchè essi siano registrati in librerie diverse.

I file registrati nella sottolibreria package o nella sottolibreria comune possono avere nomi composti da due a sette caratteri. Quelli registrati nella sottolibreria utente possono avere nomi da uno a sei caratteri. Un file della sottolibreria package deve avere un asterisco (*) come primo carattere del suo nome; un file della sottolibreria comune deve avere

un segno (+) come primo carattere del suo nome; un file della sottolibreria utente deve avere un carattere alfabetico come primo carattere del suo nome.

Per le sottolibrerie package e comune, il secondo carattere del nome di un file deve essere alfabetico. I restanti caratteri dei nomi di tutti i file (appartenenti ad una sottolibreria package, comune od utente) devono essere alfanumerici. Nel nome di un file non ci possono essere uno o più spazi interni. Tutti i caratteri alfabetici del nome di un file devono essere maiuscoli.

Vediamo alcuni esempi di nomi di file corretti e scorretti:

Sottolibreria package	- nome corretto	*SINES
	nome scorretto	* (meno di due caratteri)
Sottolibreria comune	- nome corretto	+G
	nome scorretto	+8G (secondo carattere non alfabetico)
Sottolibreria utente	- nome corretto	GRAPH2
	nome scorretto	GRAPH66 (più di 6 caratteri)

Per stampare il nome dei file presenti in una libreria si deve eseguire il comando CATALOG.

Introduzione di un comando

Un comando è composto da una parola chiave seguita, di solito, da uno o più operandi. La parola chiave (in inglese) descrive la funzione del comando. Gli operandi forniscono informazioni specifiche sulle azioni da eseguire. Le parole chiave dei comandi più usati possono essere introdotte premendo un solo tasto della sezione comandi della tastiera. Questo rende più rapida l'introduzione da tastiera del comando e riduce la possibilità di errore. Tutte le parole chiave possono essere abbreviate nei loro primi tre caratteri.

Dopo aver introdotto la parola chiave, si digitano gli operandi carattere per carattere, e si completa l'introduzione premendo il tasto [END OF LINE] . Si devono inserire uno o più spazi tra la parola chiave ed il primo operando. Dopo essere introdotto, il comando è analizzato. Se è rilevato un errore sintattico, viene imme-

diatamente visualizzato un messaggio di errore; altrimenti il comando è eseguito. Un comando non deve mai terminare con una virgola prima di (END OF LINE).

Notazioni

Nella descrizione dei comandi di sistema sono impiegate le seguenti notazioni:

{ } racchiude un insieme di parametri che non sono opzionali; uno solo di essi deve essere specificato.

[ ] racchiude un gruppo di parametri che sono opzionali; un parametro o nessun parametro può essere specificato.

_ indica un parametro assunto implicitamente dal sistema; così se si sceglie un parametro sottolineato non è necessario digitarlo.

... indica che il precedente operando può essere ripetuto più di una volta.

, separa gli operandi di un comando.

I seguenti simboli sono usati per definire il formato di un comando, ma non devono essere digitati:

- trattino di unione
_ sottolineatura
{ } parentesi graffe
[ ] parentesi quadre
... puntini

Le lettere minuscole e le parole con lettere minuscole rappresentano informazioni variabili che deve fornire l'utente.

Le lettere maiuscole, le parole con lettere maiuscole ed i seguenti simboli devono essere digitati esattamente come indicati nella definizione del comando:

segno di numero
* asterisco
+ segno più
: due punti
, virgola
= segno di uguale
/ barra

Elenco e funzione dei comandi di sistema

I comandi di sistema e le loro funzioni sono elencati in ordine alfabetico come segue. Per un elenco dei comandi per funzione si veda l'Appendice G.

Nome	Funzione
AUTO #	Genera la numerazione automatica delle linee introdotte da tastiera
CATALOG	Stampa l'indice del contenuto di una libreria
COMPILE	Converte un file testo in un programma BASIC eseguibile
CONFIGURE	Definisce particolari configurazioni logiche del sistema
CREATE	Alloca spazio in una libreria per un file dati
DATE	Registra su disco sistema o floppy disk sistema la data introdotta da tastiera per datare le operazioni sui file
DCHANGE	Permette di sostituire un floppy disk o un disco con un altro mentre il sistema è acceso, senza cancellare il contenuto della memoria principale
DECOMPILE	Converte un programma presente in memoria principale in un file testo
DELETE LINE	Cancella una o più linee di programma o di file testo presenti in memoria principale
ENVIRONMENT	Stampa le informazioni che definiscono la configurazione di sistema attualmente operante
EXEC	Carica in memoria principale ed esegue un programma di utilità (Appendice A)
FETCH	Trasferisce nel buffer di tastiera una linea di programma o di file testo presente in memoria principale
LBCLOSE	Chiude l'accesso ad una o più librerie
LBOPEN	Apre l'accesso ad una libreria specificata
LBRESTORE	Chiude le librerie attualmente aperte ed apre le librerie specificate nell'ultimo comando LBSTORE eseguito
LBSTORE	Specifica quali librerie saranno aperte ogni volta che il P6066 sarà inizializzato

LDKEYS	Assegna ai tasti funzione il contenuto registrato su disco sistema o floppy disk sistema
LINK	Inserisce un insieme di istruzioni BASIC (o una definizione di funzione), registrata in una libreria, in un programma presente nella memoria principale
LIST	Stampa una o più linee di un programma o di un file testo presente in memoria principale
LVTOC	Stampa l'elenco delle librerie presenti sull'unità a disco od a floppy disk specificata
MERGE	Fonde un file testo che è in memoria principale con uno precedentemente registrato in una libreria
MODIFY	Modifica il nome di un file e/o la dimensione di allocazione di un file dati in una libreria
NEW	Permette l'introduzione da tastiera di un programma
OLD	Carica in memoria principale un programma o file testo presente in una libreria
OPTIONS	Definisce le OPZIONI del sistema operativo da caricare in memoria principale
PREPARE	Completa l'analisi sintattica di un programma e mette il sistema nello stato di debugging
PROCEDURE	Esegue una procedura catalogata
PURGE	Cancella un file in una libreria
REPLACE	Sostituisce un programma od un file testo, presente in una libreria, con un altro presente in memoria principale senza modificarne il nome
RESEQUENCE	Modifica la numerazione delle linee del programma o del file testo presente in memoria principale
RUN	Inizia l'esecuzione di un programma
SAVE	Registra un programma od un file testo in una libreria
SECURE	Protegge un programma od un file dati impedendo la stampa, visualizzazione ed editing di una sua parte o di tutto il file

SHIFT	Modifica la numerazione delle linee del programma o file testo presente in memoria principale, iniziando da una linea specificata
SPACE	Stampa lo spazio disponibile per ulteriori registrazioni in una libreria
START	Permette di riprendere l'esecuzione di un programma dall'istruzione indicata (E' accettato nello stato di debugging.)
STKEYS	Registra su disco sistema o floppy disk sistema il contenuto assegnato ai tasti funzione
STOP	Interrompe l'esecuzione di un programma all'istruzione indicata (E' accettato nello stato di debugging.)
TEXT	Permette di introdurre un file testo da tastiera
TRANSCODE	Converte un file dati in un file testo e viceversa
TRUNCATE	Eguaglia la lunghezza di allocazione di un file dati alla lunghezza effettivamente occupata
VALIDATE	Chiude un file che è rimasto aperto in seguito alla terminazione anormale di un programma che lo apriva per operazioni di registrazione

Comando AUTO#

Funzione

Genera la numerazione automatica delle istruzioni di un programma o delle linee di un file testo.

Formato

AUT [**O#**] {**line-num**} [, **increment**]

dove:

line-num
è un numero intero positivo compreso tra 1 e 9999, che specifica il valore che sarà associato alla istruzione o linea di testo introdotta successivamente

increment
è un numero intero positivo che indica quale valore deve essere aggiunto ad ogni numero di linea per generare il numero di linea successivo.

Azione

Il comando ordina al sistema di premettere alla istruzione o linea di testo introdotta successivamente il numero di linea line-num, ed alle successive linee il numero che si ottiene aggiungendo all'ultimo generato l'incremento increment.

Ogni volta che una istruzione od una linea di testo è trasferita in memoria principale, dopo la pressione di [END OF LINE], il sistema introduce nel buffer di tastiera il numero della linea successiva che è contemporaneamente visualizzato sul display.

Il comando privo della parte opzionale ordina al sistema di iniziare la numerazione dal numero che si ottiene aggiungendo 10 al più grande numero di linea presente in memoria principale e di utilizzare un incremento di 10. Se in memoria principale non vi è presente alcuna linea la numerazione inizia da 10.

Il comando privo dell'opzione line-num ordina al

sistema di iniziare la numerazione delle linee introdotte successivamente dal numero ottenuto aggiungendo increment al più grande numero di linea presente in memoria principale. Se in memoria principale non vi è alcuna linea, la numerazione inizia dal numero specificato con increment.

Il comando privo dell'opzione increment ordina al sistema di iniziare la numerazione dal valore specificato con line-num con l'incremento di 10.

Note

1. Per interrompere la numerazione automatica si deve premere [CLEAR RECALL] con [SHIFT] ; il sistema passa nello stato comandi. Se si vuole ripristinare la numerazione automatica si deve introdurre nuovamente il comando AUTO # .

2. La parola chiave del comando può essere introdotta premendo il tasto [AUTO#] della sezione comandi.

Esempi

1. Richiedere la numerazione automatica per un nuovo programma iniziando dal numero di linea 5 con passo 20.

   Premere [A][U][T] [5][.][2][0] [END OF LINE]

2. Riprendere la numerazione automatica del seguente programma presente in memoria principale iniziando dal numero di linea 40 con passo 20. In memoria principale sono presenti le linee:

```
10 REM ZERI DI UNA FUNZIONE REALE
20 DISP " INTRODUCI I LIMITI ";
```

   Premere [AUTO#] [4][0][.][2][0] [END OF LINE]

3. Inserire istruzioni di commento nel seguente programma presente in memoria principale iniziando con il numero di linea 5 con passo 100.

```
10  ...
20  ...
30  ...
...
...
...
500 END
```

Premere [AUTO#] [5] [.] [1] [0] [0] [END OF LINE]

Si introducono le istruzioni REM volute e come risultato si ha un programma del tipo:

```
5 REM ...
10 ...
20 ...
30 ...
...
...
...
105 REM ...
...
...
...
205 REM ...
...
...
...
500 END
```

4. Richiedere la numerazione automatica di un file testo iniziando dal numero di linea 20 con passo 20.

   Premere [AUTO#] [.] [2] [0] [END OF LINE]

Comando CATALOG

Funzione

Stampa l'indice del contenuto di una libreria.

Formato

CAT[ALOG] [**filename** | * | + | :] **,[lib-ref],** [**P** | **T** | **D** | **O**] **[,F]**

dove:

filename
indica il file specificato

*
indica la sottolibreria package

+
indica la sottolibreria comune

:
indica tutte le sottolibrerie

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

P
indica file programma

T
indica file testo

D
indica file dati

F
indica che sono richieste tutte le informazioni sui file

Azione

Il comando completo di tutti gli operandi -- ad. es. CAT :,(INSTAL,UD),P,F -- ordina al sistema di stampare le seguenti informazioni per i file identificati

mediante i primi tre operandi:

- codice della Release del sistema operativo residente su disco sistema o floppy disk sistema

- etichetta di volume (accanto alla scritta VOLLABEL=. E' il nome che è stato assegnato al disco o floppy disk con il programma di utilità VOLLABEL)

- nome della libreria (accanto alla scritta LIBRARY=. E' il nome assegnato alla libreria con il programma di utilità LBCREATE)

- data (accanto alla scritta DATE:. E' l'ultima data registrata su disco sistema o floppy disk sistema con il comando DATE)

- nome del file (sotto l'intestazione FILE)

- tipo di file (sotto l'intestazione TYPE)

- data di creazione del file (sotto l'intestazione CREAT)

- data dell'ultima modifica del file (sotto l'intestazione LAST MOD)

- lunghezza di allocazione del file in byte (sotto l'intestazione MAX SIZE)

- lunghezza attuale del file in byte (sotto l'intestazione USED SIZE)

- codice di identificazione del file se appartenente ad un package fornito dalla Olivetti S.p.A. (sotto l'intestazione CODE)

- il numero di parti separate (da 1 a 4) su cui è registrato un file (sotto l'intestazione EXT)

- DEAD (se il file è danneggiato e non più accessibile)

- OPEN (se il file è un file dati rimasto aperto)

- C = XXXXX (se è un programma che utilizza un'area comune, vedi istruzione COMMON) con XXXXX numero intero di al massimo 5 cifre che specifica il numero di byte utilizzati come area comune.

Il comando privo del quarto operando, ordina al sistema di stampare le seguenti informazioni abbreviate per i file identificati mediante gli altri operandi:

- nome del file
- tipo di file

Il comando privo del terzo operando, ordina al sistema di stampare le informazioni relative a file di tipo programma, testo e dati.

Il comando privo del secondo operando, ordina al sistema di stampare le informazioni relative solo ai file della prima libreria aperta.

Il comando privo del primo operando comunica al sistema di stampare le informazioni relative ai file nella sottolibreria utente della libreria identificata con il secondo operando.

Note

1. Se il primo operando è filename, il terzo ed il quarto operando non devono essere digitati; in questo caso sono stampate tutte le informazioni riguardanti il file di nome filename.

2. L'esecuzione del comando può essere terminata premendo il tasto di console BREAK .

3. La libreria specificata con lib-name deve essere aperta.

4. Se come operando lib-ref si specifica lib-name vengono stampate le informazioni relative ai file della prima libreria aperta con nome lib-name.

5. Se come operando lib-ref si specifica (,unit-name), vengono stampate le informazioni relative ai file della prima libreria aperta residente sul disco o floppy disk a cui si riferisce il nome simbolico unit-name.

6. Le informazioni fornite dal comando CATALOG possono anche essere visualizzate su un video Olivetti DSM 6680, od un altro video, se si è eseguito un comando CONFIGURE specificando l'operando EXD od EVD.

7. Se la configurazione di sistema installata è priva di stampante ed unità video display, le informazioni fornite all'utente dal comando CATALOG sono visualizzate sul display. In questo caso, per leggere le informazioni prodotte in una linea, si devono utilizzare i tasti [→], [REPEAT] e [SHIFT], come spiegato nel cap. 1 paragrafo "Tastiera" . Per visualizzare ogni linea si deve premere il tasto CONTINUE : ad ogni pressione appare sul display una diversa linea di catalogo e quando, alla pressione di CONTINUE , il sistema produce un segnale acustico, significa che non vi sono più linee di catalogo da visualizzare.

8. Se uno o più file sono stati registrati in più parti separate di una libreria, si possono registrare su di un'unica estensione permettendo senz'altro l'impiego della parte di disco libera da informazioni significative. Si crei un'altra libreria con il programma di utilità LBCREATE e si copi in essa la libreria suddetta con il programma di utilità LIBCOPY. Si cancelli quindi la libreria di partenza eseguendo il programma di utilità LBSCRATCH e si renda libero lo spazio da essa occupato eseguendo il programma di utilità RESCTRUCT.

9. Un file è definito DEAD quando è danneggiato in conseguenza di una condizione anomala o di un'operazione sbagliata da parte dell'utente. Inviando un comando PURGE, si può cancellare il file in modo che il suo nome non compaia più nel listing ottenuto con il comando CATALOG. Il comando PURGE non consente però di ottenere lo spazio occupato dal file; per poter riavere lo spazio, è necessario seguire la seguente procedura:

   - eseguire il programma di utilità LIBCOPY per copiare il contenuto della libreria che contiene il file DEAD in un'altra libreria. Il file DEAD non viene copiato

   - eseguire il programma di utilità LBEMPTY, oppure i due programmi di utilità LBSCRATCH e LBCREATE per reinizializzare la libreria che contiene il file DEAD

   Dopo aver completato queste operazioni, si avrà una libreria che contiene i file della libreria originale (meno il file DEAD) e una seconda libre-

ria con tutto lo spazio disponibile.

10. Se un file dati è rimasto aperto, può essere richiuso eseguendo il comando VALIDATE.

Esempi

1. Si richieda la stampa del nome e del tipo di file presenti nella prima libreria INSTAL aperta.

   Premere [C][A][T] [:][.][I][N][S][T][A][L] [END OF LINE]

```
K06066-R 1.0  * VOLLABEL = SYSDIS  *  LIBRARY = INSTAL   *  DATE:10-01-80

FILE    TYPE  CREAT    LAST MOD  MAX SIZE  USED SIZE  CODE NUMBER  EXT

*PRINT   P
*TESTO   T
*DATI    R

+PROGR   P
+DATI    R
+TESTO   T

 PROGR1  P
 TESTO1  T
 DATI1   S
```

2. Si richiedano le informazioni relative al file CAL1 residente nella libreria MAT1 sul disco nell'unità con nome simbolico LD.

   Premere

   [C][A][T] [C][A][L][1][.][(][M][A][T][1][,][L][F][)] [END OF LINE]

```
CAL1    S  05-01-80  05-01-80   1024       0000
```

3. Si richieda la stampa delle informazioni complete relative ai file presenti nella sottolibreria utente della libreria MAT1 residente sul floppy disk nel trascinatore con nome simbolico FDU1.

Premere:

```
K06056-R 1.0  * VOLLABEL = SYSP60  *  LIBRARY = MAT   *  DATE:20-01-80


FILE   TYPE  CREAT    LAST MOD  MAX SIZE  USED SIZE   CODE      EXT


 CAL1    S  05-01-80  05-01-80   1024       0000                 1
 PROGR   P  05-01-80  05-01-80   0256       0256                 1
 TESTO   T  05-01-80  05-01-80   0384       0384                 1
```

Comando COMPILE

Funzione

Converte un file testo presente in memoria principale in un programma BASIC.

Formato

COM |PILE|

Azione

Il comando ordina al sistema di tradurre nel formato oggetto ogni linea del file testo, presente in memoria principale, che sia conforme alla sintassi del linguaggio BASIC. Al termine in memoria principale si ha un programma BASIC non ancora preeseguito.

Note

1. Il file testo deve essere costituito di istruzioni BASIC, altrimenti il sistema stamperà un messaggio d'errore che identificherà ciascuna linea errata. Le linee errate restano in memoria e conservano il precedente numero di linea e sono così richiamabili nel buffer di tastiera per le necessarie operazioni di editing (comando FETCH).

2. Per effettuare la preesecuzione del programma in memoria principale, è necessario inviare un comando PREPARE o RUN.

Comando CONFIGURE

Funzione

Definisce una dimensione logica della memoria utente inferiore a quella fisica effettivamente presente e permette l'impiego di periferiche esterne con funzione ausiliaria a quelle integrate.

Formato

CON[FIGURE] $\begin{bmatrix}\text{EXD}\\ \text{EVD}\end{bmatrix}$ **[,EP=n_1] [,MS=n_2] [,PRO]**

dove:

EXD

specifica che viene utilizzato un video display Olivetti DSM 6680

EVD

specifica che viene utilizzato un display video esterno (collegato mediante interfaccia RS 232-C)

n_1

è un numero intero compreso tra 8 e 15 o 24 e 31 che identifica una stampante esterna (collegata mediante interfaccia IPSO)

n_2

è un valore compreso fra 16 e 48. Il valore indica la capacità logica, in K byte, da assegnare alla memoria utente.

PRO

specifica che deve essere eseguita la procedura catalogata *SETUP ogni volta che il sistema è inizializzato.

Azione

Il comando completo di tutti gli operandi ordina al sistema di inizializzarsi configurando la memoria utente con la capacità specificata in byte con l'operando MS = n_2. Ordina inoltre che le operazioni di stampa riferite alla stampante integrata vengano eseguite sulla stampante il cui nome logico è specificato con l'operando EP = n_1 e che sullo schermo del display video o del video Olivetti DSM 6680 siano visualizzati tutti i testi che appaiono sul display integrato e sulla stampante integrata. Infine ordina che la procedura catalogata *SETUP sia eseguita ogni volta che

il sistema è inizializzato.

Se non è specificato l'operando MS, il comando CONFIGURE ordina al sistema di inizializzarsi configurando la memoria utente con la capacità fisicamente presente nella unità centrale.

Se l'operando EP noon è specificato, il comando CONFIGURE ordina al sistema che le operazioni di stampa comandate da sistema o da programma utente (istruzioni PRINT e PRINT USING) non siano più eseguibili su di una stampante IPSO ma bensì sulla stampante integrata.

Se l'operando EXD o EVD non è specificato, i testi inviati alla stampante integrata e al display integrato non vengono trasmessi al video display.

Note

1. L'esecuzione del comando CONFIGURE comporta una reinizializzazione del sistema con le modalità specificate dal comando che vengono registrate sul disco sistema o sul floppy disk sistema e rimangono valide finchè non è eseguito un nuovo comando CONFIGURE. Per stampare le informazioni che definiscono la configurazione di sistema attualmente operante, si esegua il comando ENVIRONMENT.

2. Per informazioni sull'uso del video Olivetti DSM 6680 e sulle sue prestazioni, vedere il manuale "P6066 Prestazioni grafiche - Manuale del programmatore".

3. La possibilità di configurare il sistema assegnando capacità di memoria utente minore di quella reale permette di verificare la corretta esecuzione di programmi destinati ad essere eseguiti su sistemi con capacità di memoria utente pari a quella simulata.

4. Le stampanti IPSO, PR 1350 e PR 1370 possono essere utilizzate in alternativa alla stampante integrata.

5. I caratteri che non rientrano nel set di quelli stampabili dalla stampante IPSO specificata con il comando CONFIGURE provocano la stampa del carattere ||| . Alcuni dei caratteri compresi fra i primi 32 caratteri del set ISO sono, per le stampanti IPSO, dei comandi; per cui, se uno di questi caratteri è compreso nella linea da stampare, sarà interpretato dalla stampante come comando e la stampante eseguirà le azioni ad esso relative. Gli errori relativi

alla stampante IPSO specificata nel comando CONFIGURE sono segnalati visualizzando ABN PRT, come per la stampante integrata. Perchè non ci siano errori usando una stampante IPSO tutte le operazioni di stampa devono essere completate prima che finisca il programma. Può quindi essere necessario inserire un ritardo dopo l'ultima istruzione PRINT di un programma.

6. Se viene eseguito CONFIGURE con EXD o EVD, sul video display esterno sono visualizzate le informazioni che in condizioni normali vanno solo alla stampante integrata e al display integrato.

   Più precisamente, nella prima riga dal basso sono visualizzati i messaggi emessi dal sistema e le informazioni relative alle istruzioni DISP e DISP USING, nella seconda riga è visualizzata la linea introdotta da tastiera, nella terza riga non sono visualizzati caratteri e le righe superiori del quadro presentano le informazioni normalmente destinate alla stampante integrata, rispettando la logica di funzionamento del tasto PRINT ALL.

   Se il tasto NO PRINT non è attivo, le informazioni sono presentate contemporaneamente su stampante integrata e su video display esterno. Attivando il tasto NO PRINT si blocca il funzionamento della stampante.

   Le righe più in alto della terza compongono la "pagina" del display. La pagina viene riempita dal basso verso l'alto e quando è "piena" il tasto di console CONTINUE lampeggia per avvertire l'utente che vi sono altri caratteri che compongono l'informazione da visualizzare. Premendo il tasto CONTINUE la pagina viene riempita, dal basso verso l'alto, con i rimanenti caratteri dell'informazione e i caratteri precedentemente visualizzati sono spostati verso l'alto mentre quelli che erano nella linea più in alto sono man mano sostituiti.

7. L'uso degli operandi EXD, EVD e EP fa sì che le corrispondenti routine del sistema operativo vengano caricate in memoria principale. Queste routine richiedono la seguente occupazione in Kbyte:

EXD	1 Kbyte
EVD	1 Kbyte
EP	0,5 Kbyte

Quando EXD è presente con la routine GDI (vedi comando OPTIONS) l'occupazione di memoria totale sarà di 3 Kbyte invece di 3,5 Kbyte.

8. Quando è specificato l'operando EVD, il video display che può essere utilizzato è il TCV 415V oppure qualunque video display i cui caratteri di controllo siano compatibili con esso.

9. Per informazioni sulle procedure catalogate si veda più avanti il paragrafo "PROCEDURE CATALOGATE" ed il comando PROCEDURE.

Esempio

Il seguente esempio mostra l'impiego del comando CONFIGURE per ridurre la capacità della memoria utente a 16 Kbyte.

Si prema [C][O][N] [M][S] = [1][6]

Comando CREATE

Funzione

Alloca, in una libreria, lo spazio necessario a contenere un file dati.

Formato

CRE[ATE] filename, [lib-ref], [S|R|Z] [,n]

dove:

filename
indica il nome di un file

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

S
indica file dati di tipo sequenziale

R
indica file dati ad accesso diretto inizializzato con dati numerici in singola precisione che non hanno un valore assegnato

Z
indica file dati ad accesso diretto inizializzato con dati numerici in singola precisione il cui valore è zero

n
indica il numero di byte assegnati al file dati.

Azione

Il comando completo con tutti gli operandi ordina al sistema di allocare, per il file dati di nome filename,

n byte nella libreria specificata e di registrarne il nome nel catalogo della libreria.

Il comando privo del quarto operando ordina al sistema di allocare, per il file dati di nome filename, 4096 byte nella libreria specificata.

Note

1. In una libreria non vi possono essere due file con lo stesso nome.

2. Non si possono "creare" nuovi file dati in una sottolibreria package protetta.

3. Non si possono creare in una sottolibreria più file di quanti dichiarati per essa durante l'esecuzione del programma di utilità LBCREATE (vedi appendice A).

4. Il numero di byte richiesto viene arrotondato al successivo multiplo di 128 (per file su floppy disk) o di 256 (per file su disco).

5. La libreria specificata con lib-name deve essere aperta.

6. Se come operando lib-ref è specificato lib-name, lo spazio richiesto per il file è allocato nella prima libreria aperta di nome lib-name.

7. Se lib-name è omesso lo spazio richiesto per il file è allocato nella prima libreria aperta sul disco o floppy disk identificato dal nome simbolico unit-name.

8. Se l'operando lib-ref è omesso lo spazio richiesto per il file è allocato sulla prima libreria aperta.

Esempi

1. Si crei un file dati ad accesso diretto di 1024 byte nella sottolibreria comune della libreria MAT residente nell'unità UD.

   Premere

   CRE +CALL.(MAT.UD).
   R.1024 END OF LINE

2. Si crei un file dati sequenziale di 4096 byte nella sottolibreria utente della prima libreria aperta.

Premere [C] [R] [E] [D] [A] [T] [A] [END OF LINE]

Comando DATE

Funzione

Permette di datare le operazioni sui file registrati nelle librerie.

Formato

DAT [E] date

dove:
date
è una sequenza di 6 caratteri che indica una data.

Azione

Il comando comunica al sistema di registrare sul disco sistema o sul floppy disk sistema la data introdotta da tastiera come operando del comando stesso.

Si devono sempre introdurre 6 caratteri e nessuno di essi può essere lo spazio.

Note

1. Durante l'inizializzazione del sistema, (dopo l'accensione o durante l'esecuzione dei comandi OPTIONS e CONFIGURE) la sequenza di caratteri registrata per ultima sul disco sistema o sul floppy disk sistema è caricata in memoria principale e quindi utilizzata per datare le operazioni sui file dati esterni.

2. La sequenza introdotta sarà, in generale: ggmmaa (giorno, mese ed anno).

Esempi

1. Si registri la data: 1 gennaio 1980.

   Premere D A T 0 1 1 0 8 0 END OF LINE

Comando DCHANGE

Funzione

Permette di sostituire un disco, od un floppy disk, mentre il sistema è in funzione, senza cancellare il contenuto della memoria principale.

Formato

DCH[ANGE] unit-name

dove:
unit- name
indica il nome simbolico di una unità a disco od a floppy disk.

Azione

Il comando comunica al sistema che si vuol sostituire il disco, o floppy disk, presente nella unità con nome simbolico unit-name.

Quando il comando è eseguito vengono chiuse tutte le librerie che erano aperte sul disco o floppy disk che deve essere sostituito ed è visualizzato il messaggio:

LOAD DISK ON unit-name

per avvertire l'utente di sostituire il disco, o floppy disk, presente nell'unità con nome simbolico unit-name. Dopo aver inserito il nuovo disco, o floppy disk, si deve premere il tasto di console CONTINUE.

Se la prima libreria presente nel disco inserito, o la libreria presente nel floppy disk inserito, non è stata creata associando ad essa una parola d'ordine e se vi sono meno di 6 librerie già aperte attualmente è visualizzato il messaggio:

LIBR. lib-name OPEN ON UNIT unit-name

per avvertire l'utente che la libreria con nome simbolico lib-name è stata aperta dal sistema.

Note

1. Se si sostituisce un disco sistema ad un floppy disk sistema, il nuovo disco o floppy disk deve contenere la stessa release di sistema operativo e le stesse caratteristiche e funzioni (selezionate durante la generazione di sistema).

2. Il comando deve essere usato ogni qual volta si vuole sostituire un disco od un floppy disk con un altro oppure inserire un secondo disco, mentre la macchina è accesa.

3. Se si apre lo sportello di un trascinatore di una unità a disco, senza prima aver introdotto il comando DCHANGE, il sistema visualizza il messaggio: ABN unit-name - DCH OMITTED, quando esegue una operazione di accesso al disco presente nel relativo trascinatore. L'esecuzione dell'eventuale programma sarà terminata.

Esempi

1. Si introduca un floppy disk in sostituzione di un altro già presente nell'unità con nome simbolico FDU1, il cui spazio libero sia insufficiente a contenere un programma presente in memoria principale.

   Premere D C H F D U 1 END OF LINE

   Introdurre il disco e quindi premere CONTINUE .

2. Si sostituisca il disco presente nell'unità con nome simbolico UD, senza spegnere il P6066.

   Premere D C H U D END OF LINE

   Dopo aver sostituito il disco, si prema CONTINUE .

Comando DECOMPILE

Funzione

Converte un programma BASIC presente in memoria principale in un file testo.

Formato

DEC [OMPILE]

Azione

Il comando comunica al sistema di convertire il programma presente in memoria principale in un file testo.

Note

1. Non si possono convertire in file testo i programmi che sono stati protetti mediante il comando SECURE.

2. Le linee del file testo hanno un formato diverso dalle linee introdotte durante la generazione del programma, poichè il sistema esegue operazioni di editing sulle linee durante l'esecuzione del comando DECOMPILE.

3. Il comando DECOMPILE è utile nel caso in cui in un programma sono state cancellate alcune variabili o richiami di funzione, o riferimenti a linee di programma (es. istruzioni GOTO), durante la fase di editing, perchè tali riferimenti permangono allo interno del sistema ma sono definitivamente cancellati se il programma è decompilato mediante il comando DECOMPILE e poi compilato con il comando COMPILE.

Comando DELETE LINE

Funzione

Cancella una o più linee di programma o di testo presenti in memoria principale.

Formato

DEL [ETE LINE] [line-num$_1$[, line-num$_2$]]

dove:

line-num$_1$
indica il numero della linea da cancellare o la prima linea di un insieme di linee da cancellare

line-num$_2$
indica l'ultima linea di un insieme di linee da cancellare.

Azione

Il comando, completo di tutti gli operandi, comunica al sistema di cancellare le linee del programma o del file testo presente in memoria principale comprese tra i numeri di linea indicati con il primo e secondo operando (estremi inclusi).

Il comando, privo del secondo operando, comunica al sistema di cancellare la linea specificata con il primo operando.

Il comando, privo di operandi, comunica al sistema di cancellare una delle seguenti linee:

- l'ultima linea corretta introdotta da tastiera
- l'ultima linea visualizzata con il comando FETCH od i tasti [↑] e [↓]
- l'ultima linea stampata mediante un comando LIST
- l'ultima linea di un programma eseguito mediante un comando RUN

Note

1. Non si possono cancellare le linee di un programma protette con il comando SECURE.

2. La parola chiave DELETE LINE può essere introdotta premendo il tasto [DELETE LINE] insieme con [SHIFT].

3. Se durante la fase di editing di un programma BASIC si cancellano delle variabili, dei richiami di funzioni definite dall'utente o dei richiami ad altre istruzioni di programma, tali riferimenti permangono all'interno del sistema; per cancellarli definitivamente si decompili il programma con il comando DECOMPILE e poi lo si compili con il comando COMPILE.

Esempi

1. Si cancellino le istruzioni di un programma, presente in memoria principale, dal numero di linea 50 al numero di linea 150.

   Premere [D][E][L] [5][0][.][1][5][0] [END OF LINE]

2. Si cancelli la linea 100 del file testo presente in memoria principale.

   Premere [SHIFT][DELETE LINE] [1][0][0] [END OF LINE]

Comando ENVIRONMENT

Funzione

Stampa le informazioni che definiscono la configurazione di sistema attualmente operante.

Formato

ENV [IRONMENT]

Azione

Il comando comunica al sistema di stampare le seguenti informazioni:

- codice della release di sistema operativo residente su disco sistema o floppy disk sistema
- data
- capacità della memoria principale
- governi di I/O presenti nel P6066
- situazione dichiarata dall'ultimo comando CONFIGURE eseguito
- opzioni richieste dall'ultimo comando OPTIONS eseguito
- nome simbolico delle unità a disco e floppy disk, nell'ordine NAME (nome), PERIPH CODE (codice interno dell'unità) e PERIPH TYPE (tipo di unità)
- su quale disco risiede il sistema operativo
- elenco delle librerie aperte quando è eseguito il comando
- elenco delle librerie che sono aperte ogni qual volta si inizializza il sistema

Esempio

Si richieda la stampa delle informazioni che definiscono la configurazione di sistema presente.

Premere E N V END OF LINE

```
K05066-R 1.0 *        SYSTEM ENVIRONMENT      * DATE:10-01-80

            MEMORY PHYSICAL SIZE : 48 K
            I/O INTERFACES : IPSO-1
                             DCU

            MEMORY LOGICAL SIZE: 24 K
            EXTERNAL PRINTER :  NO
            EXTERNAL VIDEO DISPLAY :  NO
            SOFTWARE OPTIONS : STRING,MATR,EXT INTPT

            DISK UNITS:

            NAME        PERIPH CODE        PERIPH TYPE
            LD          A0                 DCU
            UD          A1                 DCU
            FDU1        C0                 FDU
            FDU2        C1                 FDU

            OPERATING SYSTEM ON DISK UNIT LD

OPEN LIBRARIES :
                    NAME          DISK UNIT
                    TLX           LD
                    PROV          UD
                    P6FSYS        FDU2
                    INSTAL        FDU1
STORED LIBRARIES :
                    NAME          DISK UNIT
                    TLX           LD
                    PROV          UD
```

Si noti che la capacità di memoria utente è stata ridotta da 48K a 24K eseguendo un comando CONFIGURE con MS = 24.

Comando EXEC

Funzione

Carica in memoria ed esegue un programma di utilità.

Formato

EXE [C] utility [, parameter [, parameter]...]

dove:
utility
è il nome di un programma di utilità
parameter
specifica un operando associato con il programma di utilità specificato.

Azione

Il comando comunica al sistema di caricare in memoria principale ed eseguire il programma di utilità di nome utility.

Nota

1. Nella libreria di software di base sono disponibili i seguenti programmi utilità:

   DCOPY
   DINIT
   FLCOPY
   FLPRINT
   LBCREATE
   LBEMPTY
   LBPROTECT
   LBRENAME
   LBSCRATCH
   LIBCOPY
   RESTRUCT
   VOLLABEL

   Per ulteriori informazioni si veda l'appendice A.

Comando FETCH

Funzione

Trasferisce nel buffer di tastiera una linea di programma o di file testo presente in memoria principale.

Formato

FET [CH] [line-num]

dove:
line-num
è il numero di linea della linea da trasferire nel buffer di tastiera.

Azione

Il comando comunica al sistema di trasferire nel buffer di tastiera e visualizzare sul display la linea del programma o file testo presente in memoria principale con numero line-num.

Il comando senza operandi comunica al sistema di trasferire nel buffer di tastiera e di visualizzare sul display una delle seguenti linee:

- l'ultima linea corretta introdotta da tastiera
- l'ultima linea di un file caricato in memoria principale mediante il comando OLD
- l'ultima linea stampata con il comando LIST
- l'ultima istruzione di un programma appena eseguito ed ancora presente in memoria principale
- l'ultima linea richiamata nel buffer di tastiera con il comando FETCH o con i tasti [↑] e [↓].

Note

1. Se si introduce il comando FETCH con un operando line-num il cui valore non è presente in memoria principale, la linea con il numero di linea imme-

diatamente inferiore a quella specificata viene trasferita nel buffer di tastiera. Se line-num è inferiore al più piccolo numero di linea presente in memoria principale, nel buffer di tastiera viene trasferita la linea con il più grande numero di linea.

2. La linea visualizzata sul display non ha lo stesso formato di quella introdotta da tastiera, perchè il sistema la modifica prima di trasferirla nel buffer di tastiera; così, ad esempio, quando si introduce la istruzione: 10A=B il comando FETCH la visualizzerà sul display come: 0010 LET A=B. Se l'editing effettuato dal sistema produce una linea con più di 80 caratteri, la linea viene compattata eliminando tutti gli spazi.

3. Se l'eliminazione degli spazi non significativi non riporta la linea ad 80 caratteri viene visualizzato un messaggio di errore e la linea è stampata sulla stampante integrata. Premendo il tasto di console RECALL la linea appare sul display (nel buffer di tastiera è troncata ai primi 80 caratteri) e si può editare; quando è introdotta nuovamente, premendo il tasto END OF LINE, sostituisce la linea precedente che era presente, integralmente, nel working-file.

4. Non si può usare il comando FETCH riferito a linee di un programma protette mediante il comando SECURE.

5. La parola chiave può essere introdotta premendo il tasto [FETCH NEW] con [SHIFT].

Esempi

1. Trasferire nel buffer di tastiera la linea 50 di un programma presente in memoria principale.

   Premere [F][E][T] [5][0] [END OF LINE]

2. Trasferire nel buffer di tastiera l'ultima linea di un file testo caricato in memoria principale con il comando OLD.

   Premere [SHIFT][FETCH] [END OF LINE]

Comando LBCLOSE

Funzione

Chiude l'accesso ad una o più librerie.

Formato

LBC [LOSE] { lib-ref / * }

dove:

lib-ref

può essere uno dei seguenti operandi:

(lib-name, unit-name)

lib-name
(,unit-name)

con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

*

indica che devono essere chiuse tutte le librerie attualmente aperte.

Azione

Il comando comunica al sistema di chiudere la libreria specificata con lib-name, residente nel disco o floppy disk presente nell'unità con nome simbolico unit-name.

Il comando con lib-ref = (,unit-name), comunica al sistema di chiudere tutte le librerie aperte residenti nell'unità con nome simbolico unit-name.

Il comando con lib-ref = lib-name, comunica al sistema di chiudere la prima libreria aperta di nome lib-name.

Il comando con * come operando comunica al sistema di chiudere tutte le librerie attualmente aperte.

Nota

Eseguendo il comando LVTOC con l'operando * si possono stampare i nomi delle librerie attualmente aperte.

Esempio

Si chiuda la libreria MAT residente nella unità con nome simbolico LD.

Premere [L][B][C] [(][M][A][T][,][L][D][)] [END OF LINE]

Comando LBOPEN

Funzione

Apre l'accesso ad una libreria specificata.

Formato

LBO [PEN] lib-ref [, password]

dove:

lib-ref

può essere uno dei seguenti operandi:

(lib-name, unit-name)
lib-name
(,unit-name)

con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

password

è una stringa di al massimo 6 caratteri uguale alla stringa password specificata quando è stata creata la libreria con il programma di utilità LBCREATE o uguale alla parola d'ordine di sistema

Azione

Il comando comunica al sistema di aprire la libreria specificata con lib-name, residente nell'unità con nome simbolico unit-name.

Se come lib-ref è specificato lib-name, viene aperta la libreria di nome lib-name residente su disco sistema o su floppy disk sistema.

Se come lib-ref è specificato (,unit-name), viene aperta la prima libreria residente nell'unità con nome simbolico unit-name.

Note

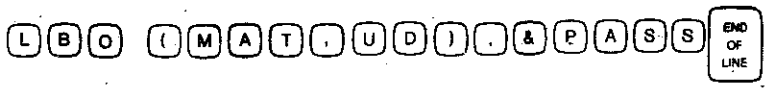
1. Se la libreria è stata creata associando ad essa una parola d'ordine, questa deve essere specificata quando si introduce il comando.

2. Eseguendo il comando VTOC con l'operando * si possono stampare i nomi delle librerie attualmente aperte.

Esempio

Si apra la libreria MAT residente nell'unità UD avente &PASS come parola d'ordine.

Premere



Comando LBRESTORE

Funzione

Riapre le librerie specificate nell'ultimo comando LBSTORE eseguito e chiude quelle attualmente aperte.

Formato

LBR[ESTORE]

Azione

Il comando ordina al sistema di chiudere le librerie attualmente aperte e di aprire, successivamente, le librerie specificate nell'ultimo comando LBSTORE eseguito.

Note

1. Se il comando LBSTORE non è mai stato eseguito, l'esecuzione del comando LBRESTORE apre la prima libreria residente sul disco sistema o la libreria residente su floppy disk sistema, se questa è stata creata senza assegnarle una parola d'ordine.

2. Per sapere quali librerie sono aperte dopo che è stato eseguito il comando LBRESTORE, si esegua il comando LVTOC con l'operando *.

Comando LBSTORE

Funzione

Memorizza le librerie da aprire ogni volta che il sistema P6066 sarà inizializzato.

Formato

LBS [TORE]

Azione

Il comando LBSTORE richiede al sistema di riaprire automaticamente tutte le librerie attualmente aperte ogni qual volta il sistema P6066 è inizializzato (accensione, esecuzione del comando OPTIONS o del comando CONFIGURE). Dopo che il comando è eseguito, il sistema stampa il nome delle librerie che devono essere aperte, nell'ordine con cui saranno aperte.

Note

1. Se si esegue LBSTORE senza che vi siano librerie aperte, o se non si esegue mai LBSTORE, ad una successiva inizializzazione del sistema P6066 non viene aperta alcuna libreria e si ha una segnalazione di errore.

2. Il comando LVTOC permette di conoscere quali sono le librerie che vengono aperte ogni qual volta il sistema è inizializzato e con quale ordine sono aperte.

3. Se una libreria protetta da una parola d'ordine è aperta quando si esegue un comando LBSTORE, quella libreria sarà poi aperta automaticamente senza bisogno della parola d'ordine. Per evitare che questo succeda, si deve per prima cosa chiudere la libreria per mezzo del comando LBCLOSE e quindi inviare il comando LBSTORE.

Esempio

Si esegua il comando LBSTORE.

Premere L B S END OF LINE

```
LBS
NAME          PERIPHERAL
LIBR3         LD
INSTAL        LD
LIB1          UD
MAT           LF
```

Come si vede viene stampato il nome delle librerie da aprire ogni qual volta il sistema è inizializzato. L'ordine con cui sono stampati i nomi coincide con l'ordine con cui le relative librerie saranno aperte.

Comando LDKEYS

Funzione

Riassegna ai tasti funzione il contenuto assegnato precedentemente ed attualmente memorizzato sul disco sistema o su floppy disk sistema.

Formato

LDK [EYS]

Azione

Il comando comunica al sistema di riassegnare ad ogni tasto funzione la stringa di caratteri che ad esso era già stata assegnata in precedenza ed era stata successivamente registrata sul disco sistema o sul floppy disk sistema da un comando STKEYS.

Note

1. Il comando è usato perchè il contenuto dei tasti funzione può essere modificato da programma, mediante l'istruzione FKEY #, oppure durante gli stati esecuzione calcoli immediati e debugging mediante FKEY # (vedi capitoli 6 e 7).

2. Il contenuto presente su disco sistema o su floppy disk sistema è riassegnato ai tasti funzione ogni volta che il sistema è inizializzato, quindi: (1) quando il sistema è acceso, (2) quando si esegue il comando OPTIONS e (3) quando si esegue il comando CONFIGURE.

Esempio

Si registri sul disco sistema o sul floppy disk sistema un contenuto per i tasti funzione F1, F2, F3. Dopo aver assegnato ai medesimi tasti funzione un contenuto diverso, si ripristini il precedente.

Premere CALC MODE

Il sistema è nello stato esecuzione di calcoli immediati.

Premere [SHIFT] [FKEY#] [1] [,] [R] [E] [P] [L] [A] [C] [E] [END OF LINE]

[SHIFT] [FKEY#] [2] [,] [P] [U] [R] [G] [E] [END OF LINE]

[SHIFT] [FKEY#] [3] [,] [O] [L] [I] [V] [E] [T] [T] [I] [P] [6] [0] [6] [6] [END OF LINE]

ai tasti funzione F1, F2, F3 sono assegnati i caratteri a destra della virgola.

Premere [CALC MODE]

Il sistema è nello stato comandi.

Premere [S] [T] [K] [END OF LINE]

La suddetta associazione fra tasti funzione e relativo contenuto è registrata sul disco sistema o sul floppy disk sistema

Premere [CALC MODE]

Premere [SHIFT] [FKEY#] [1] [,] [R] [U] [N] [D] [E] [M] [O] [END OF LINE]

[SHIFT] [FKEY#] [2] [,] [R] [E] [S] [E] [Q] [U] [E] [N] [C] [E] [END OF LINE]

[SHIFT] [FKEY#] [3] [,] [L] [I] [N] [K] [END OF LINE]

Si modifica il contenuto dei tasti funzione F1,F2,F3.

Premere [CALC MODE]

Il sistema è di nuovo nello stato comandi.

Premendo [F1] sul display si vede: RUN DEMO
Premendo [F2] sul display si vede: RESEQUENCE
Premendo [F3] sul display si vede: LINK

Premere [L] [D] [K] [END OF LINE]

Premendo [F1] sul display si vede: REPLACE
Premendo [F2] sul display si vede: PURGE
Premendo [F3] sul display si vede: OLIVETTI P6066

Come si vede è stato ripristinato il contenuto dei tasti funzione che era stato registrato sul disco sistema o sul floppy disk sistema.

Comando LINK

Funzione

Inserisce un insieme di istruzioni BASIC od una definizione di funzione, registrati su una libreria come file testo, in un programma presente in memoria principale.

Formato

LIN[K] filename, [lib-ref], line-num [,α]

dove:

filename
è il nome di un file testo contenente linee di programma BASIC o una definizione di funzione

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

line-num
indica il numero di linea che sarà assegnato alla prima linea del testo all'atto dell'inserimento nel programma presente in memoria principale

α
è una lettera maiuscola che completa il nome con cui sarà chiamata la definizione di funzione inserita nel programma

Azione

Il comando, con l'operando α , ordina al sistema di aggiungere al programma presente in memoria principale la definizione di funzione registrata nella libreria lib-name, residente nell'unità con nome simbolico unit-name, come file testo col nome filename. Il primo numero di linea della definizione di funzione inserita nel programma dal comando LINK è uguale al

valore specificato con line-num; i numeri di linea successivi mantengono l'incremento che essi avevano nel file testo presente nella libreria suddetta. Il sistema assegna alla definizione di funzione il nome FNα o, nel caso di una definizione di funzione di tipo stringa, FNα\$. Al termine della esecuzione del comando LINK, in memoria principale si ha un nuovo programma.

Il comando, senza l'operando α, ordina al sistema di inserire nel programma presente in memoria principale l'insieme di istruzioni BASIC registrato come file testo di nome filename nella libreria lib-name, residente nell'unità con mome simbolico unit-name. Le istruzioni inserite sono rinumerate a partire dal numero di linea line-num, mantenendo fra un numero di linea ed il successivo lo stesso incremento già presente nel file file-name. Al termine della esecuzione del comando, in memoria principale si ha un nuovo programma.

Note

1. Il numero di linea specificato con line-num non deve essere già presente in memoria principale.

2. Se il numero di linea più alto del file da inserire in memoria principale è n e l'operando line-num ha valore m, allora in memoria principale non devono già essere presenti numeri di linea compresi tra m ed m+n.

3. Il comando LINK non può essere eseguito se il programma presente in memoria principale è stato protetto mediante il comando SECURE.

4. E' bene verificare che dopo l'esecuzione del comando LINK il programma presente in memoria principale abbia una ed una sola istruzione END con il più alto numero di linea.

5. Se come operando lib-ref è specificato lib-name, il sottoprogramma o la definizione di funzione, di nome filename, è prelevato dalla prima libreria aperta di nome lib-name.

6. Se come operando lib-ref è specificato (,unit-name), il sottoprogramma o la definizione di funzione, di nome filename, è prelevato dalla prima libreria aperta residente nell'unità con nome simbolico

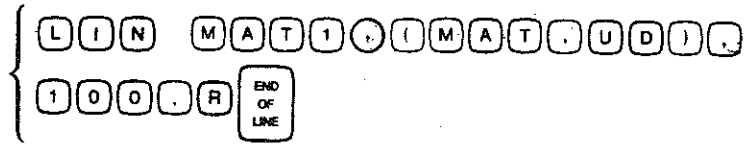
unit-name.

7. Se l'operando lib-ref non è specificato, il sottoprogramma o la definizione di funzione, di nome filename, viene ricercato nelle librerie attualmente aperte secondo l'ordine di apertura.

Esempi

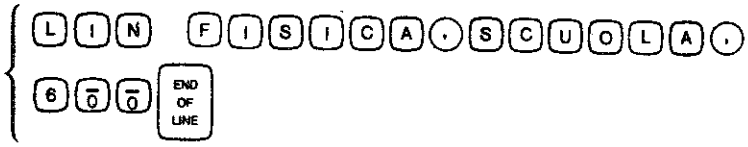
1. Si aggiunga ad un programma presente in memoria principale (iniziando da line-num = 100) una definizione di funzione numerica MAT1 registrata, come file testo, nella libreria MAT residente nell'unità con nome simbolico UD. Alla definizione di funzione si assegni il nome FNR.

   Premere

2. Inserire un sottoprogramma, registrato come file testo di nome FISICA nella prima libreria aperta di nome SCUOLA, in un programma presente in memoria principale iniziando da line-num uguale a 600.

   Premere

Comando LIST

Funzione

Stampa una o più linee di un programma o file testo presente in memoria principale.

Formato

LIS[T] [line-num$_1$] $\left[,\left\{\begin{matrix}\textbf{[line-num}_2\textbf{]}\ \textbf{,X} \\ \textbf{line-num}_2\end{matrix}\right\}\right]$

dove:

line-num$_1$
indica la linea da stampare o la prima linea di un insieme di linee da stampare

line-num$_2$
indica l'ultima linea di un insieme di linee da stampare

X
indica che non devono essere stampati i numeri di linea di un file testo.

Azione

Il comando ordina al sistema di stampare le linee del programma o del file testo presente in memoria principale con numero di linea compreso tra il numero di linea indicato con il primo operando e quello indicato con il secondo operando (estremi inclusi).

Il comando, privo degli ultimi due operandi, ordina al sistema di stampare le linee del programma o del file testo presente in memoria principale iniziando dalla linea il cui numero di linea è specificato con il primo operando.

Il comando, privo del primo operando, ordina al sistema di stampare le linee il cui numero di linea è inferiore o uguale a quello specificato.

Il comando, senza operandi, ordina al sistema di stampare tutte le linee del programma o del file testo presente in memoria principale.

Con l'operando X specificato, il comando ordina al sistema di stampare le linee specificate del file testo presente in memoria principale, senza il numero di linea.

Note

1. Poichè le istruzioni di un programma sono editate durante la stampa eseguita con il comando LIST, le linee specificate nel comando possono essere stampate con un formato diverso da quello con cui sono state introdotte. Per esempio, se si introduce 40A=B, il sistema stampa ØØ40 LET A=B. Se tale editing produce una linea con più di 80 caratteri, allora la linea viene compattata eliminando gli spazi.

2. Lo stesso editing è attuato sulle linee di testo di un file testo prodotto mediante un comando DECOMPILE partendo da un programma.

3. Il comando LIST non stampa le linee di un programma protette mediante il comando SECURE.

4. La parola chiave del comando LIST può essere introdotta premendo il tasto [LIST].

5. Il comando LIST può visualizzare le linee di un programma o di un file testo su un video Olivetti DSM 6680 o su un altro video se si è eseguito un comando CONFIGURE specificando l'operando EXD o EVD.

6. Se la configurazione di sistema installata è priva di stampante e di unità video display, le linee di programma o di file testo sono visualizzate sul display. In questo caso, per leggere una linea completa, si devono utilizzare i tasti [→], [REPEAT] e [SHIFT] come spiegato nel cap. 1, § "Tastiera". Per leggere ogni linea si deve premere il tasto [CONTINUE] : ad ogni pressione appare sul display una diversa linea e quando, alla pressione di [CONTINUE], il sistema produce un segnale acustico, significa che non vi sono più linee da visualizzare.

Esempi

1. Si richieda la stampa delle linee di un file testo presente in memoria principale. Non si stampino i numeri di linea.

Premere [LIST] [,] [,] [X] [END OF LINE]

2. Si richieda la stampa di una istruzione di programma presente in memoria principale con il numero di linea 45.

   Premere [L] [I] [S] [T] [4] [5] [,] [4] [5] [END OF LINE]

Comando LVTOC

Funzione

Stampa alcune informazioni relative alle librerie presenti su una unità a disco (od a floppy disk) o l'elenco delle librerie aperte attualmente e quello delle librerie aperte ogni qual volta il sistema è inizializzato.

Formato

LVT[OC] [**unit-name** | *]

dove:

unit-name
è il nome simbolico associato ad una unità a disco od a floppy disk

*
indica di stampare l'elenco delle librerie attualmente aperte e di quelle memorizzate come librerie da aprire a ogni inizializzazione del sistema

default
vengono stampate alcune informazioni relative alle librerie residenti su disco sistema o su floppy disk sistema.

Azione

Il comando ordina al sistema di stampare le seguenti informazioni, relative ad ogni libreria residente sul disco o floppy disk nell'unità con nome simbolico <u>unit-name</u>:

- nome della Release del sistema operativo, con l'identificatore di numero di Release e di livello, ad esempio: KØ 6066-R 1.0 dove 1.Ø identifica la prima distribuzione di livello Ø

- etichetta di identificazione del supporto fisico (volume), stampata come VOLLABEL = XXXXXX dove XXXXXX è l'etichetta (vedi in appendice A il programma di utilità VOLLABEL)

- il formato della traccia, ossia il numero di byte che compongono ogni settore segnalato come: TRACK FORMAT = nnn BYTE, dove nnn è 256 nel caso di disco e 128 nel caso di floppy disk

- la data registrata con il comando di sistema DATE, stampata accanto alla stringa DATE:

- per ogni libreria è stampato nell'ordine: il nome della libreria, la data di creazione, la posizione di inizio della libreria sul disco (nel formato TTT SSS, con TTT numero di traccia ed SSS numero di settore), la posizione di fine della libreria sul disco nel formato TTT SSS), il numero di settori che sono allocati per contenere i file della libreria, il numero di settori che sono ancora liberi per la registrazione di altri file nella libreria

- il numero di settori che sono liberi per la creazione di nuove librerie, utilizzabili dopo aver eseguito il programma di utilità RESTRUCT che riordina lo spazio su disco dedicato alle librerie, nel formato FREE SPACE (TOTAL) = nnnnnn SECTORS

- il numero di settori che sono liberi per la creazione di nuove librerie senza riorganizzare la struttura delle informazioni sul disco, nel formato MAXIMUM BLOCK OF FREE SPACE = nnnnnn SECTORS.

Il comando LVTOC senza operandi ordina al sistema di stampare le suddette informazioni per le librerie residenti sul disco sistema o sul floppy disk sistema.

Se il comando è specificato con * come operando, è stampato l'elenco delle librerie attualmente aperte, nell'ordine di apertura, e l'elenco delle librerie memorizzate come librerie da aprire a ogni inizializzazione del sistema, nell'ordine di apertura. Accanto al nome delle suddette librerie è stampato il nome simbolico dell'unità in cui risiede la libreria.

Esempi

1. Si stampino le informazioni relative alle librerie residenti sul disco nell'unità con nome simbolico UD.

   Premere [L] [V] [T] [U] [D] [END OF LINE]

Il sistema stampa:

```
K06066-R 1.0 * VOLLABEL=UPPER * TRACK FORMAT=256 BYTE * DATE:10-01-80

LIBRARY  CREAT     BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
LIB3     02-11-77  001000        001039        40         32
LIB1     02-11-77  001040        002031        40         27
FREE SPACE (TOTAL) = 19072 SECTORS
MAXIMUM BLOCK OF FREE SPACE   = 19072 SECTORS
```

2. Si stampi l'elenco delle librerie attualmente aperte e di quelle aperte quando il sistema è inizializzato.

   Premere [L] [V] [T] [*] [END OF LINE]

   Il sistema stampa:

```
OPEN LIBRARIES
                    NAME        DISK UNIT
                    LIBR3       LD
                    INSTAL      LD
                    LIB1        UD
                    MAT         LF

STORED LIBRARIES
                    NAME        DISK UNIT
                    LIBR3       LD
                    INSTAL      LD
                    LIB1        UD
                    MAT         LF
```

Comando MERGE

Funzione

Effettua la fusione di un file testo in memoria principale con uno precedentemente registrato in una libreria.

Formato

MER[GE] filename, [lib-ref], [line-num], [/]

dove:

filename
specifica il nome di un file testo presente in una libreria

lib-ref
può essere uno dei seguenti 3 operandi:
(lib-name, unit-name)
lib-name
(,unit name)
dove lib-name specifica il nome di una libreria aperta e unit-name specifica il nome simbolico di un disco o di una unità floppy disk

line-num
è il numero di linea da assegnare alla prima linea del filename

/
richiede al sistema di non fornire una stampa di tutte le linee sostituite.

Azione

Il comando MERGE, se specificato con tutti gli operandi, effettua la rinumerazione del file filename presente nella libreria specificata iniziando dal numero di linea line-num e mantenendo l'incremento originale. Successivamente il file testo filename viene aggiunto al testo che è in memoria principale. Se una linea del file filename e una linea del testo in memoria hanno lo stesso numero, la linea nuova sostituirà quella vecchia. Il risultato sarà un file testo aggiornato e modificato in memoria principale.

Se MERGE è specificato senza il quarto operando, /, il sistema fornirà la stampa di ogni linea sostituita insieme alla linea che la sostituisce. Il listing avrà il seguente formato:

```
####    PREVIOUS LINE (linea precedente)
####    PRESENT LINE  (linea attuale)
```

dove ####è un numero di linea.

Se MERGE non specifica l'operando line-num, il file filename non viene rinumerato.

Note

1. Se viene specificato (lib-name, unit-name) come operando di lib-ref, il file testo richiesto sarà quello nella libreria specificata sull'unità identificata da unit-name.

2. Se viene specificato soltanto il nome di una libreria come operando di lib-ref, la ricerca del file testo richiesto incomincia dalla prima libreria aperta che ha il nome specificato.

3. Se viene specificato (,unit-name) come operando di lib-ref, la ricerca del file testo richiesto inizia dalla prima libreria aperta sull'unità identificata da unit-name.

4. Se viene omesso lib-ref, la ricerca del file testo richiesto inizia dalla prima libreria aperta e continua nell'ordine in cui sono state aperte le librerie.

5. Il file testo filename è rinumerato solamente in memoria principale mentre rimane invariato nella libreria.

Esempi

1. Un file testo TEX1, registrato sull'unità FDU1 nella libreria P6FSYS, viene fuso ad un file testo che è in memoria principale con il nome TEX2.
   Per chiarire meglio l'operazione di fusione, i due file testo sono stati listati per prima cosa.
   Il comando MERGE richiede al sistema di rinumerare il file TEX1 prima di fare la fusione (iniziando dal numero di linea 85). Si vede il risultato della fusione nel listing che segue il comando inviato.

```
OLD TEX1


LIST
FILE      TEX1


0010 TEXT 1
0020 TEXT 1
0030 TEXT 1
0040 TEXT 1
0050 TEXT 1
0060 TEXT 1
0070 TEXT 1
0080 TEXT 1
0090 TEXT 1
0100 TEXT 1

END OF LISTING


OLD TEX2
LIST
FILE      TEX2


0010 TEXT 2
0020 TEXT 2
0030 TEXT 2
0040 TEXT 2
0050 TEXT 2
0060 TEXT 2
0070 TEXT 2
0080 TEXT 2
0090 TEXT 2
0100 TEXT 2
0110 TEXT 2
0120 TEXT 2
0130 TEXT 2

END OF LISTING


MER TEX1,(P8FSYS,FDU1),85

LIST
FILE      TEX2


0010 TEXT 2
0020 TEXT 2
0030 TEXT 2
0040 TEXT 2
0050 TEXT 2
0060 TEXT 2
0070 TEXT 2
0080 TEXT 2
0085 TEXT 1
0090 TEXT 2
0095 TEXT 1
0100 TEXT 2
0105 TEXT 1
0110 TEXT 2
0115 TEXT 1
0120 TEXT 2
0125 TEXT 1
0130 TEXT 2
0135 TEXT 1
0145 TEXT 1
0155 TEXT 1
0165 TEXT 1
0175 TEXT 1

END OF LISTING
```

2. In questo esempio si vede che al file testo TEX1 precedente è assegnato come numero di linea della prima linea 70. Durante la fusione, le linee da 70 a 130 del file TEX2 vengono sostituite dalle linee del file TEX1 con lo stesso numero di linea. Avendo specificato l'operando /, il sistema non ci fornirà un listing che mostra la sostituzione.

```
OLD TEX2


LIST
FILE      TEX2

0010 TEXT 2
0020 TEXT 2
0030 TEXT 2
0040 TEXT 2
0050 TEXT 2
0060 TEXT 2
0070 TEXT 2
0080 TEXT 2
0090 TEXT 2
0100 TEXT 2
0110 TEXT 2
0120 TEXT 2
0130 TEXT 2

END OF LISTING





MER TEX1,(P6FSYS,FDU1),70,/


LIST
FILE      TEX2

0010 TEXT 2
0020 TEXT 2
0030 TEXT 2
0040 TEXT 2
0050 TEXT 2
0060 TEXT 2
0070 TEXT 1
0080 TEXT 1
0090 TEXT 1
0100 TEXT 1
0110 TEXT 1
0120 TEXT 1
0130 TEXT 1
0140 TEXT 1
0150 TEXT 1
0160 TEXT 1

END OF LISTING
```

3. Questo esempio fa vedere cosa succede se viene omesso l'operando /. Ci verrà dato un listing di tutte le linee sostituite.

```
OLD TEX2

LIST
FILE      TEX2

0010 TEXT 2
0020 TEXT 2
0030 TEXT 2
0040 TEXT 2
0050 TEXT 2
0060 TEXT 2
0070 TEXT 2
0080 TEXT 2
0090 TEXT 2
0100 TEXT 2
0110 TEXT 2
0120 TEXT 2
0130 TEXT 2

END OF LISTING

MER TEX1,(P&FSYS,FDU1),50

REPLACED LINES:
#### PREVIOUS LINE
#### PRESENT LINE

0050 TEXT 2
0050 TEXT 1

0060 TEXT 2
0060 TEXT 1

0070 TEXT 2
0070 TEXT 1

0080 TEXT 2
0080 TEXT 1

0090 TEXT 2
0090 TEXT 1

0100 TEXT 2
0100 TEXT 1

0110 TEXT 2
0110 TEXT 1

0120 TEXT 2
0120 TEXT 1

0130 TEXT 2
0130 TEXT 1

LIST
FILE      TEX2

0010 TEXT 2
0020 TEXT 2
0030 TEXT 2
0040 TEXT 2
0050 TEXT 1
0060 TEXT 1
0070 TEXT 1
0080 TEXT 1
0090 TEXT 1
0100 TEXT 1
0110 TEXT 1
0120 TEXT 1
0130 TEXT 1
0140 TEXT 1

END OF LISTING
```

Comando MODIFY

Funzione

Modifica il nome di un file e/o il numero di byte riservati in una libreria ad un file dati.

Formato

MOD [IFY] old-filename, [lib-ref], {new-filename [, n] / ,n}

old-filename
indica il nome di un file programma, testo o dati

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

new-filename
indica il nuovo nome da dare al file programma, testo o dati

n
è un numero che indica il numero di byte da riallocare nella libreria per il file specificato

Azione

Il comando, completo di tutti gli operandi, ordina al sistema di assegnare al file dati registrato con il nome old-filename nella libreria lib-name, sul disco contenuto nella unità con nome simbolico unit-name, il nome new-filename e di riservare ad esso n byte nella libreria lib-name.

Il comando, senza l'operando n, ordina al sistema di sostituire al file programma, testo o dati registrato nella libreria lib-name con il nome old-filename il nome new-filename.

Il comando, con n come terzo operando e senza l'operando new-filename, ordina al sistema di riservare n byte al file dati registrato nella libreria lib-name

con il nome old-filename.

Note

1. L'operando new-filename può essere costituito solamente da un massimo di 6 caratteri alfanumerici di cui il primo alfabetico.

2. In una stessa libreria non possono esistere file con lo stesso nome.

3. Non si può modificare il nome di un file della sottolibreria package o della sottolibreria comune dopo che sono state protette con il programma di utilità LBPROTECT.

4. Si possono modificare i nomi dei file di una sottolibreria package o comune prima che siano state protette, specificando come new-filename una stringa di al massimo 6 caratteri alfanumerici di cui il primo alfabetico. Il sistema aggiungerà automaticamente * o +.

5. Se n non è multiplo di 256, nel caso di disco o di 128, nel caso di floppy disk, il sistema rialloca per il file dati specificato un numero di byte pari al successivo multiplo di 256 o di 128.

6. Per un file dati sequenziale n può essere minore della dimensione di allocazione specificata in precedenza, ma non può essere minore della sua dimensione attuale.

7. Per un file ad accesso diretto n non può essere minore della sua dimensione di allocazione.

8. La libreria contenente il file old-filename deve essere aperta.

9. Se come lib-ref è specificato solamente lib-name, il file interessato alla modifica è cercato nella prima libreria aperta, in ordine di apertura, di nome lib-name.

10. Se come lib-ref è specificato (,unit-name), il file interessato alla modifica è cercato nella prima libreria aperta, in ordine di apertura, sull'unità con nome simbolico unit-name.

11. Se l'operando lib-ref non è specificato, il file

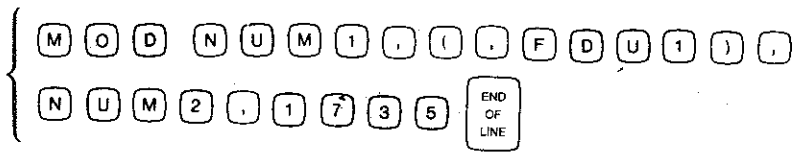
interessato alla modifica è cercato nelle librerie aperte secondo l'ordine di apertura.

12. Il comando MODIFY non può essere usato con programmi o file dati protetti mediante il comando SECURE.

Esempi

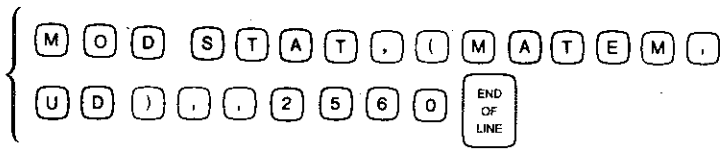
1. Modificare il nome del file dati NUM1 in NUM2 e riallocare 1735 byte per esso sul floppy disk con nome simbolico FDU1.

   Premere

2. Modificare lo spazio riservato al file dati STAT, nella libreria MATEM residetne sul disco nell'unità con nome simbolico UD, da 4096 byte a 2560 byte.

   Premere

Comando NEW

Funzione

Predispone il sistema ad accettare l'introduzione di un programma da tastiera.

Formato

NEW

Azione

Il comando ordina al sistema di allocare spazio in memoria principale per permettere di introdurre un programma, linea per linea, da tastiera.

Note

1. L'esecuzione del comando NEW cancella il precedente contenuto della memoria utente; quindi, prima di introdurre NEW, si deve introdurre SAVE o REPLACE per registrare il programma o file testo, eventualmente presente in memoria principale, in una libreria.

2. Dopo l'inizializzazione o reinizializzazione (comando OPTIONS e CONFIGURE) del sistema, si può introdurre un programma da tastiera senza digitare il comando NEW.

3. Introducendo il comando AUTO # si può avere la numerazione automatica delle linee introdotte successivamente.

Esempio

Generare un nuovo programma da tastiera mentre in memoria principale esiste già un programma.

Premere [SHIFT] [SAVE] [+] [M] [A] [T] [END OF LINE]

così il programma è registrato col nome +MAT nella prima libreria (è inserito nella sottolibreria comune).

Premere [NEW] [END OF LINE]

Ora si può introdurre il nuovo programma, linea per linea.

Comando OLD

Funzione

Carica in memoria principale un programma o file testo registrato in una libreria.

Formato

OLD filename [,lib-ref]

filename
è il nome del programma o file testo da caricare in memoria principale

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

Azione

Il comando ordina al sistema di caricare in memoria principale il programma o file testo filename, registrato nella libreria lib-name, residente sul disco o floppy disk della unità unit-name.

Note

1. La libreria specificata con lib-name deve essere aperta.

2. Se come operando lib-ref è specificato solamente lib-name il file filename è cercato nella prima libreria aperta, in ordine di apertura, di nome lib-name.

2. Se come operando lib-ref è specificato (,unit-name) il file filename è cercato nella prima libreria aperta, in ordine di apertura, residente sul disco nella unità con nome simbolico unit-name.

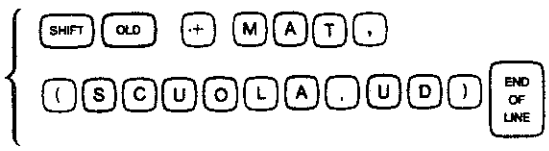
4. Se l'operando lib-ref non è specificato il file <u>filename</u> è cercato nelle librerie aperte secondo l'ordine con cui sono state aperte.

5. La parola chiave del comando può essere introdotta premendo i tasti [SHIFT] e [OLD] contemporaneamente.

Esempio

Caricare in memoria principale il programma +MAT, presente nella sottolibreria comune, della libreria SCUOLA, residente sul disco dell'unità con nome simbolico UD.

Premere



Comando OPTIONS

Funzione

Definisce le opzioni del sistema operativo che devono essere caricate in memoria principale ogni volta che il sistema è inizializzato.

Formato

OPT [IONS] [option [,option]...]

dove:
option
Può essere DEB, MAT, STR, RS232, XNT o alternativamente PLO o GDI; nessuna opzione può essere ripetuta e vi possono essere al massimo 6 operandi. Le opzioni possono essere specificate in qualsiasi ordine.

Azione

Il comando ordina al sistema di inizializzarsi caricando in memoria le routine del sistema operativo specificate dagli operandi.

L'operando DEB permette l'esecuzione del debugging di un modulo assembler richiamato da un programma BASIC: in particolare sono caricate in memoria utente le routine di sistema operativo relative che occupano 3,5K byte (K = 1024). Per ulteriori informazioni si veda la pubblicazione:"P6066 Ambiente di programmazione in Assembler",codice 3978380 T).

L'operando MAT permette di eseguire programmi contenenti istruzioni BASIC di calcolo matriciale: in particolare vengono caricate in memoria utente le routine di sistema operativo relative che occupano 1,5K byte.

L'operando STR permette di eseguire programmi contenenti istruzioni BASIC che impiegano le funzioni BLN$ e REP$: in particolare vengono caricate in memoria utente le routine del sistema operativo relative che occupano 1K byte.

L'operando RS232 permette di eseguire programmi contenenti istruzioni BASIC di input-output che indirizzano periferiche esterne collegate al sistema mediante interfaccia seriale (EIA RS232-C). In particolare vengono caricate in memoria utente le relative routine del sistema operativo che occupano 1 Kbyte.

L'operando XNT richiede che il sistema sia inizializzato in modo che l'utente possa gestire le interruzioni che arrivano da unità periferiche esterne. Le routine caricate per la funzione occupano 1 Kbyte di memoria utente.

L'operando PLO richiede che si possano eseguire programmi contenenti istruzioni BASIC che permettano di tracciare immagini utilizzando come plotter la stampante integrata. In particolare vengono caricate in memoria utente le routine del sistema operativo relative che occupano 2K byte.

L'operando GDI richiede che il sistema sia inizializzato in modo che si possa plottare su uno schermo video Olivetti DSM 6680 tramite un programma in BASIC. Le routine caricate per la funzione occupano 2,5K byte di memoria utente. Se viene caricata anche la routine per la funzione EXD (vedi CONFIGURE) lo spazio complessivo occupato sarà di 3K byte invece di 3,5K byte.

Il comando, senza operandi, comunica al sistema di cancellare dalla memoria utente le routine del sistema operativo corrispondenti alle opzioni eventualmente presenti.

Quando viene specificato solo con qualche operando, OPTIONS richiede al sistema che cancelli dalla memoria utente le routine precedentemente caricate e che carichi quelle specificate.

Note

1. L'insieme di opzioni richiesto da ogni comando OPTIONS è memorizzato su disco sistema o floppy disk sistema così che ogni volta che il sistema viene acceso ricarica le opzioni richieste dall'ultimo comando OPTIONS eseguito.

2. Quando si esegue il comando OPTIONS, il precedente contenuto nella memoria principale è cancellato. Se si vuole registrare il contenuto della memoria utente, prima di introdurre il comando OPTIONS, si

introduca il comando SAVE od il comando REPLACE.

3. Se è specificato PLO, non si deve specificare GDI e viceversa.

4. E' possibile introdurre un programma che tratta le matrici, che traccia grafici o impiega le periferiche seriali, anche se la configurazione del sistema non ne permette l'esecuzione. Tale programma può essere registrato in una libreria con il comando SAVE ed eseguito dopo aver introdotto il comando OPTIONS con le opzioni necessarie.

5. Per sapere quali opzioni sono attualmente attive si esegua il comando ENVIRONMENT.

6. Per ulteriori informazioni si vedano i manuali: "P6066 I/O con periferiche esterne" e "P6066 Prestazioni grafiche".

Esempi

1. Reinizializzare il sistema per poter eseguire programmi che elaborano matrici e che tracciano grafici con la stampante integrata.

   Premere [0] [P] [T] [M] [A] [T] [,] [P] [L] [0] [END OF LINE]

2. Cancellare dalla memoria utente le routine opzionali del sistema operativo.

   Premere [0] [P] [T] [END OF LINE]

Comando PREPARE

Funzione

Esegue la preesecuzione di un programma ed al termine pone il sistema nello stato di DEBUGGING.

Formato

PRE[PARE] [filename [,lib-ref]]

dove:
filename
è il nome di un programma registrato in una libreria
lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

Azione

Il comando ordina al sistema di caricare in memoria principale il programma con nome filename, registrato nella libreria lib-name residente sul disco o floppy disk nell'unità con nome simbolico unit-name, e di eseguirne la preesecuzione.

Il comando, privo di operando, ordina al sistema di eseguire la preesecuzione del programma presente in memoria principale.

La preesecuzione di un programma BASIC comporta l'esecuzione di diverse operazioni tra le quali:

1. Controllo della corretta compilazione delle funzioni multilinea.

2. Controllo della correttezza dei riferimenti ad al-

tre istruzioni contenuti nelle istruzioni dei cicli FOR/NEXT.

3. Controllo della correttezza dei riferimenti ad altre istruzioni contenuti nelle istruzioni delle funzioni multilinea.

4. Controllo della correttezza delle istruzioni di salto.

5. Controllo della posizione nel programma delle istruzioni FOR rispetto alle istruzioni NEXT e delle istruzioni DEF rispetto alle istruzioni FNEND.

6. Posizionamento del pointer del file dati interno sul primo dato.

7. Caricamento delle routine del sistema operativo necessarie per l'esecuzione del programma.

8. Allocazione di spazio per le variabili.

9. Allocazione di buffer per i file dati esterni riferiti nel programma.

10. Apertura dei file dati riferiti nel programma.

11. Allocazione di buffer per l'impiego di periferiche esterne.

Se dopo l'esecuzione del comando non è segnalato alcun errore, il programma presente in memoria è in formato eseguibile ed il sistema è nello stato di debugging. Viene emesso il messaggio: *** FORMALLY CORRECT PROGRAM *** che è stampato se PRINTALL è attivo. Viene inoltre stampato il messaggio:
:: ROOM = X ::
dove X è il numero di byte ancora disponibile in memoria utente per l'esecuzione del programma. Si osservi che durante l'esecuzione del programma il sistema utilizza questa parte di memoria utente per cui si possono avere segnalazioni di spazio di memoria insufficiente a seconda delle caratteristiche del programma utente stesso. Infine sul display compare il messaggio: PROGRAM nome-programma READY TO RUN. Premendo il tasto BREAK il sistema commuta nello stato comandi.

Se dopo l'esecuzione del comando PREPARE non è stato segnalato alcun errore, si può eseguire il programma presente in memoria principale premendo il tasto CONTINUE ; oppure premendo il tasto STEP si può eseguire una istruzione per volta. Prima di comandare l'esecuzione del programma si possono inizializzare le variabili con valori assegnati da tastiera.

Se durante la preesecuzione si rilevano degli errori, vengono stampati i relativi messaggi di errore. Il sistema è nello stato comandi e si possono effettuare le necessarie correzioni.

Note

1. Se come operando lib-ref è specificato lib-name, il programma di nome filename è cercato nella prima libreria aperta di nome lib-name.

2. Se come operando lib-ref è specificato (, unit-name), il programma di nome filename è cercato nella prima libreria aperta residente sul disco o floppy disk nella unità con nome simbolico unit-name.

3. Se l'operando lib-ref non è specificato il programma di nome filename è cercato nelle librerie aperte secondo l'ordine di apertura.

4. Quando un programma è stato preeseguito mediante il comando PREPARE od un comando RUN (vedi RUN) una successiva esecuzione comandata con un comando RUN evita la fase di preesecuzione. Se il programma è registrato su disco o su floppy disk il sistema ricorda che è stato preeseguito, per cui ogni successiva esecuzione comandata con il comando RUN evita la fase di preesecuzione. Se in un programma preeseguito si eseguono azioni di editing, il programma deve essere nuovamente preeseguito.

5. Per poter eseguire un comando PREPARE senza gli operandi si deve avere sulla prima libreria aperta uno spazio libero da registrazioni equivalente a quello occupato dal programma in memoria principale.

6. Se un programma è registrato in una libreria senza che sia prima preeseguito sarà preeseguito ogni qualvolta se ne comanderà l'esecuzione con il comando RUN. Per evitare la perdita di tempo dovuta

alla preesecuzione si richiami il programma in memoria principale eseguendo il comando OLD, si preesegua il programma eseguendo un RUN o PREPARE e quindi si esegua il comando REPLACE.

Esempio

Completare l'analisi di coerenza tra le istruzioni di un programma presente in memoria principale; si assegnino quindi alle sue variabili dei valori durante lo stato di debugging. Il programma presente in memoria è:

```
10 PRINT A$;B,C$;D
20 A$="AREA"
30 C$="RETTANGOLO"
40 PRINT A$+C$; B*D
50 END
```

Premere P R E END OF LINE

Il sistema è nello stato di debugging. Da tastiera si assegnano i valori:

```
A$="BASE="
C$="ALTEZZA="
 B=10
 D=100
```

Premere CONTINUE

Il programma è eseguito e stampa:

```
BASE= 10          ALTEZZA= 100
AREA RETTANGOLO 1000
```

Generalità

Una sequenza di comandi di sistema o programmi BASIC, od una combinazione di comandi e programmi, che siano utilizzati di frequente, possono essere registrate in una libreria come procedure catalogate e quindi eseguite come singole unità. L'impiego di una procedura catalogata può abbreviare il tempo totale di esecuzione dei corrispondenti lavori che richiederebbero altrimenti un intervento continuo dell'operatore attraverso la tastiera.

Una procedura catalogata è introdotta da tastiera come un file testo, quindi trascodificata in un file dati sequenziale e registrata in una libreria mediante un comando TRANSCODE. Al termine di queste operazioni, può essere eseguita mediante un comando PROCEDURE. Come primo passo per creare una procedura si introduca il comando TEXT. Poichè ogni linea della procedura inizia con un numero di linea -- ed i numeri devono essere in ordine crescente -- si può successivamente introdurre il comando AUTO# . Il modo con cui si codificano le linee di una procedura dipende dal tipo di procedura che si genera. In generale:

1. Se la procedura si compone di soli comandi, si digiti un comando per volta anteponendo ad esso un numero di linea.

2. Se la procedura è una sequenza di programmi, si introduca una istruzione per linea -- compreso il numero di linea dell'istruzione. Così ogni linea di ogni programma avrà due numeri: il primo corrispondente alla sua posizione nella procedura, il secondo alla sua posizione nel programma. Ogni programma scritto per la procedura deve essere preceduto da un comando NEW e deve terminare con una istruzione END. Vediamo un semplice esempio:

```
0010 NEW
0020 10 PRINT"PROGRAMMA 1"
0030 20 END
0040 RUN
0050 NEW
0060 10 PRINT"PROGRAMMA 2"
0070 20 END
0080 RUN
```

Si noti che quando una linea contiene due numeri essi devono essere separati da almeno uno spazio.

3. Se una procedura è una combinazione di programmi e comandi, valgono le regole espresse al punto 2. Inoltre, si osservi che non è necessario che il comando RUN per l'esecuzione di un programma segua immediatamente l'istruzione END. Esso può essere posto dopo alcuni comandi che seguono l'istruzione END. Tuttavia, tali comandi non possono essere:

   CONFIGURE
   NEW
   OPTIONS
   PROCEDURE filename
   RUN filename
   TEXT

   Ognuno di questi comandi cancellerebbe il contenuto della memoria principale e quindi renderebbe non valido il comando RUN.

La seguente è un esempio di procedura composta da programmi e comandi:

```
0010 NEW
0020 10 PRINT"PROGRAMMA 1"
0030 20 END
0040 LIST
0050 RUN
0060 RUN NUMBER,LIB2
0070 NEW
0080 10 PRINT"PROGRAMMA 2"
0090 END
0100 LIST
0110 ENVIRONMENT
0120 RUN
```

Si noti che poichè una procedura è registrata come file dati, essa può essere creata mediante un programma BASIC che registri le linee della procedura in forma di stringhe di caratteri in un file dati sequenziale.

Input automatici

I valori richiesti dalle istruzioni INPUT che appaiono sia nei programmi richiamati da una procedura (RUN filename) che in quelli scritti come parte di una procedura,possono essere introdotti automaticamente. Utilizzando questa possibilità, si può evitare l'interruzione dell'esecuzione di un programma e la necessità di introdurre dati da tastiera ogni qualvolta è eseguita un'istruzione INPUT.

Si può predisporre l'input automatico per ogni comando RUN, contenuto in una procedura, che esegue un programma contenente istruzioni INPUT. La prestazione è predisposta facendo seguire immediatamente al comando RUN una o più linee con il seguente formato:

$$\text{IN} = \left\{\begin{array}{l}\text{num-constant}\\ \\ \text{string-constant}\end{array}\right\}, \left[\left\{\begin{array}{l}\text{num-constant}\\ \\ \text{string-constant}\end{array}\right\}\right] \ldots$$

Una linea di tale formato deve essere scritta per ogni istruzione INPUT nel programma corrispondente. I valori specificati a destra del segno uguale sono assegnati automaticamente alle variabili specificate in ogni istruzione INPUT corrispondente, quando è eseguito un comando PROCEDURE che ha come operando A. (Si noti che la stringa IN = non deve contenere uno spazio).

Si osservi che il numero di linee IN = deve essere uguale al numero di istruzioni INPUT del corrispondente programma. Se vi sono più linee IN = di quelle richieste, le linee in eccesso sono ignorate; ma se il numero di linee IN = è inferiore a quello richiesto, viene visualizzato il seguente messaggio d'errore:

ERROR 204 IN LINE line-number

Se, invece, il numero di valori specificati in una istruzione IN = è superiore al numero di variabili della corrispondente istruzione INPUT, i valori in eccesso sono semplicemente ignorati.

Questo messaggio d'errore viene inviato anche se viene data una risposta non valida a una richiesta di input, come ad esempio se si impostano dati stringa invece di dati numerici oppure se si imposta un numero insufficiente di valori per un'istruzione INPUT.

Salti condizionati

Una procedura può contenere linee che ne modificano l'ordine di esecuzione se è soddisfatta una certa condizione. L'esecuzione della procedura riprende da una linea specificata se il programma eseguito in precedenza ha assegnato ad una variabile CC un valore predefinito. La sintassi di una linea di salto condizionato è:

IF CC = X, line-number

dove X è un valore da 0 a 9 e line-number è il numero di linea di una linea della procedura. Il valore a CC è assegnato, dal programma eseguito in precedenza, per mezzo di una variabile stringa (con lunghezza di allocazione di 1) specificata come primo valore nell'area comune (vedi nel capitolo 5 l'istruzione COMMON).

Procedura avviata alla inizializzazione del sistema

Per creare una procedura catalogata che sarà eseguita automaticamente ogni volta che il sistema è acceso o, comunque, inizializzato (comandi CONFIGURE ed OPTIONS), si eseguano i seguenti passi:

1. Si apra una libreria (se non vi sono librerie aperte) e si registri in essa la procedura presente in memoria principale, eseguendo un comando TRANSCODE con l'operando filename uguale a *SETUP.

2. Si esegua il comando LBSTORE per fare in modo che la libreria contenente *SETUP sia aperta quando il sistema è inizializzato.

3. Si esegua un comando CONFIGURE con l'operando PRO.

Test di una procedura: comando TEST

Il comando TEST permette di controllare l'esecuzione di una procedura per: (1) determinare se vi sono degli errori, (2) procedere nell'esecuzione anche se vi sono degli errori e (3) conoscere il numero totale degli errori. Il comando TEST è descritto qui piuttosto che in una parte ad esso specifica, perchè può essere

usato unicamente in una procedura catalogata.

Il formato del comando è:

$$\text{TEST} \left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \\ \# \end{array} \right\}$$

La prestazione è attivata specificando: TEST ON nel punto della procedura in cui si vuole che inizi il controllo, ad esempio, subito dopo il comando PROCEDURE o subito prima di una serie di comandi di cui si vuol controllare la validità. Subito dopo l'esecuzione di TEST ON, il sistema stampa il codice d'errore di ogni errore che si verifica durante l'esecuzione dei comandi successivi. La procedura sarà eseguita fino alla fine anche se si verifica un errore di programma, ma in questo caso l'esecuzione del programma è interrotta ed è eseguito il comando successivo della procedura.

Per terminare l'effetto prodotto dal comando TEST ON si deve specificare nella procedura un comando TEST OFF o TEST# . Gli errori che si verificano dopo l'esecuzione di TEST OFF o TEST# e prima che sia eseguito un successivo TEST ON sospendono l'esecuzione della procedura catalogata. La differenza tra gli ultimi due comandi sta nel fatto che quando è eseguito TEST# il sistema stampa il numero totale di errori che si è verificato fino a quel momento.

Esempio

Per un esempio che illustri come si crea una procedura catalogata si veda il comando PROCEDURE.

Comando PROCEDURE

Funzione

Inizia l'esecuzione di una procedura catalogata.

Formato

PRO [CEDURE] filename, [lib-ref], [A], [line-num$_1$] [,line-num$_2$]

dove:

filename
specifica il nome di una procedura catalogata (registrata in una libreria come file dati sequenziale)

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
dove lib-name specifica il nome di una libreria e unit-name specifica il nome simbolico di una unità a disco o floppy disk

A
specifica l'impiego degli input automatici

line-num$_1$
specifica la linea della procedura da cui deve iniziare l'esecuzione

line-num$_2$
specifica l'ultima linea che deve essere eseguita

Azione

Quando è specificato con tutti gli operandi, il comando PROCEDURE ordina al sistema di eseguire la procedura catalogata filename, registrata nella libreria specificata da lib-ref, iniziando dalla linea line-num$_1$ e finendo con la linea line-num$_2$. I valori alle variabili delle istruzioni INPUT di tutti i programmi contenuti nella procedura, o richiamati dalla procedura, saranno assegnati automaticamente per mezzo delle linee IN = comprese nella procedura.

Quando line-num$_1$ non è specificato nel comando PROCEDURE, l'esecuzione inizia dalla prima linea della procedura.

Quando line-num$_2$ non è specificato nel comando PROCEDURE, l'esecuzione continua finchè non è eseguita l'ultima linea della procedura.

Note

1. Se come operando lib-ref è specificato (lib-name, unit-name), la procedura deve essere nella libreria lib-name sul disco o floppy disk nell'unità il cui nome simbolico è unit-name.

2. Se come operando lib-ref è specificato lib-name, la procedura è cercata nella prima libreria aperta di nome lib-name.

3. Se come operando lib-ref è specificato (,unit-name), la ricerca della procedura specificata inizia dalla prima libreria aperta residente sull'unità con nome simbolico unit-name.

4. Se l'operando lib-ref è omesso, la ricerca della procedura inizia dalla prima libreria aperta e continua nell'ordine con cui le librerie sono aperte.

5. L'operando line-num$_1$ non può essere il numero di linea di un'istruzione BASIC.

6. Se la funzione PRINT ALL è attiva, ogni linea della procedura eseguita è visualizzata e stampata.

7. Quando l'esecuzione di una procedura è completa, il sistema commuta nello stato comandi.

8. Si può terminare l'esecuzione di una procedura premendo BREAK. Il sistema commuta nello stato comandi.

9. Se una linea di una procedura ha un errore di sintassi (ad esempio LITT invece di LIST), è visualizzato un messaggio di errore ed il sistema commuta nello stato comandi. Introducendo il comando:

   TRANSCODE T, filename, lib-ref,#

   (dove filename è il nome della procedura), la pro-

cedura è caricata in memoria principale come file testo per le correzioni necessarie. Dopo che la correzione è stata effettuata, la versione modificata della procedura può essere registrata di nuovo, ed essere disponibile per successive esecuzioni, introducendo i comandi:

PURGE filename, lib-ref
TRANSCODE D, filename, lib-ref,#

10. Se durante l'esecuzione di un programma che fa parte di una procedura è visualizzato un errore recuperabile, l'esecuzione è interrotta ed il sistema commuta nello stato di debugging. Dopo aver premuto BREAK si può correggere l'errore eseguendo le operazioni descritte nel punto 9. Oppure si può continuare l'esecuzione del programma, e quindi della procedura, premendo CONTINUE .

11. Se durante l'esecuzione di una procedura è visualizzato un errore non recuperabile riferito ad un programma della procedura, l'esecuzione del programma è interrotta. Il tasto di console STEP è acceso per informare l'utente che può verificare il contenuto delle variabili del programma, come quando il sistema è nello STATO DI DEBUGGING. Dopo aver eventualmente verificato il contenuto delle variabili di programma, tuttavia, si deve premere BREAK . Il sistema commuta nello STATO COMANDI. Per correggere il programma, si eseguono le operazioni descritte nel punto 9.

12. Se si introduce un comando PROCEDURE senza l'operando A ma nella relativa procedura vi sono linee IN =, tali linee sono ignorate.

13. Se una procedura contiene un comando OPTIONS o CONFIGURE, l'esecuzione della procedura termina dopo l'esecuzione del comando.

14. Una procedura può richiedere l'esecuzione di altre procedure mediante comandi PROCEDURE.

15. Se durante l'esecuzione di una procedura è eseguita una linea del tipo IF CC = x, line-number con line-number più grande del numero di linea $line\text{-}num_2$ specificato nel comando PROCEDURE, l'esecuzione della procedura riprende dal numero di linea

line-number. L'esecuzione della procedura terminerà quando sarà eseguita la linea della procedura con il più grande numero di linea o quella con numerc di linea line-num$_2$.

16. Quando il sistema è nello STATO COMANDI a causa di un errore incontrato durante l'esecuzione di una procedura, si può riprendere l'esecuzione della procedura introducendo un comando PROCEDURE senza operandi. La prima linea eseguita sarà quella che segue la linea errata. Questo metodo di lavoro è comunque sconsigliabile perchè può produrre dei risultati imprevedibili.

Esempio

Vediamo un esempio di procedura, PRO1, composta da tre programmi, alcuni comandi e due linee IF.

```
0010 OLD P1
0020 LIST
0030 RUN
0040 IN=3,5,10
0050 IF CC=0,80
0060 RUN P3
0070 IF CC=1,90
0080 RUN P2
0090 PRO STOP
```

Questa procedura permette di calcolare alcuni valori della funzione $\frac{1}{X-A}$ definita in un intervallo $[X(2),X(3)]$ tale che $X(2) < X(3)$ e che in esso vi sia il punto di ascissa A.

Il programma P1 è composto dalle seguenti istruzioni:

```
0010 DCL 1A$
0020 COMMON A$,X()
0030 DIM X(3)
0040 INPUT X(1),X(2),X(3)
0050 IF X(2)>X(3) THEN 100
0060 IF (X(2)<X(1)) AND (X(3)>X(1)) THEN 100
0070 IF (X(2)=X(1))OR(X(3)=X(1)) THEN 100
0080 A$="1"
0090 GOTO 110
0100 A$="0"
0110 END
```

ed è stato registrato nella libreria MAT, specificando l'impiego di un'area comune di 28 byte, con il seguente comando:

SAV P1, MAT,, COM = 28

Mediante l'istruzione COMMON, linea 20, i valori assegnati ad A$, X(1), X(2) ed X(3) sono registrati in una area comune. Il valore corrispondente al punto A è registrato nel secondo elemento dell'area comune X(1). Il programma quindi verifica che X(2) < X(3) e che A sia compreso nell'intervallo definito. Se tali condizioni sono soddisfatte un 1, nel formato stringa, è registrato nel primo elemento dell'area comune. Se le condizioni non sono soddisfatte, viene registrato uno Ø nel formato stringa. Se le condizioni non sono soddisfatte, la linea 50 della procedura legge lo Ø e cede il controllo dell'esecuzione alla linea 80, che provoca l'esecuzione del programma P2.

Il programma P2 è composto dalle seguenti due istruzioni:

```
0010 PRINT "VALORI INTRODOTTI NON VALIDI"
0020 END
```

Se le condizioni testate dal programma P1 sono soddisfatte, la linea 50 della procedura passa il controllo all'esecuzione del programma P3. Il programma P3 è composto dalle seguenti istruzioni:

```
0010 DCL 1A$
0020 DIM X(3)
0030 COMMON A$,X()
0040 DEF FNA(X,A)=1/(X-A)
0050 FOR I=X(2) TO X(3) STEP 0.1
0060 PRINT I,FNA(I,X(1))
0070 NEXT I
0080 END
```

ed è anch'esso registrato specificando l'impiego di una area comune di 28 byte. Come si vede vengono stampati i valori assunti dalla funzione $\frac{1}{X-A}$ per valori variabili, con incremento di 0,1, nell'intervallo [X(2),X(3)].

Dopo che uno dei programmi P2 o P3 è eseguito, viene eseguita la procedura STOP che è composta dalle seguenti linee:

```
0010 OLD TEXT,MAT
0020 LIST ,,X
```

con il file testo TEXT composto dalle seguenti linee:

```
0010 Olivetti P6066
0020 PROCEDURA - PRO1
```

Per creare la procedura PRO1, avendo in precedenza creato e registrato su disco (o floppy disk) la procedura STOP ed i programmi P1, P2 e P3, essa deve prima essere introdotta in memoria come file testo e quindi trascodificata in un file dati eseguendo il comando:

TRA D, PRO1, MAT

Se si esegue la procedura con il comando:

PRO PRO1,,A

che richiede l'esecuzione di input automatici, i valori assegnati alle variabili X(1), X(2) ed X(3) saranno nell'ordine 3, 5 e 10.

Se invece la procedura PRO1 è eseguita con il comando:

PRO PRO1

i valori alle variabili suddette potranno essere assegnati da tastiera quando l'esecuzione del programma P1 sarà interrotta per effetto dell'esecuzione dell'istruzione INPUT nella linea 40.

Comando PURGE

Funzione

Cancella un file da una libreria.

Formato

PUR[GE] filename [, lib-ref]

dove:
filename
è il nome di un file da cancellare
lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

Azione

Il comando ordina al sistema di cancellare il file specificato con l'operando filename dalla libreria lib-name sul disco o floppy disk nell'unità con nome simbolico unit-name. Lo spazio occupato dal file viene reso disponibile per altre operazioni sulla stessa libreria.

Se è specificato il solo operando filename, il file da cancellare è cercato nelle librerie attualmente aperte secondo l'ordine di apertura.

Note

1. Se come operando lib-ref è specificato lib-name, il file da cancellare è cercato nella prima libreria aperta di nome lib-name.

2. Se come operando lib-ref è specificato (,unit-name), il file da cancellare è cercato nella prima libreria aperta residente sul disco o floppy disk nella unità con nome simbolico unit-name.

3. Il comando non è accettato con i file che appartengono alla sottolibreria package ed alla sottolibreria comune, se queste sono state protette con il programma di utilità LBPROTECT.

4. Quando si invia il comando PURGE per cancellare un file definito DEAD, il file viene cancellato solo logicamente; lo spazio, infatti, resta occupato. Vedere il comando CATALOG per ulteriori spiegazioni sulla riallocazione dello spazio.

Esempio

Si cancelli il programma MAT1 dalla libreria MAT presente sul disco nell'unità UD.

Premere [P][U][R] [M][A][T][1][(][M][A][T][,][U][D][)] [END OF LINE]

Comando REPLACE

Funzione

Sostituisce, in una libreria, un programma od un file testo con un altro presente in memoria principale.

Formato

REP [LACE] [lib-ref]

dove:
lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

Azione

Il comando ordina al sistema di registrare nella libreria lib-name, sul disco o floppy disk nell'unità con nome simbolico unit-name, il programma o file testo presente in memoria principale al posto del programma o file testo già registrato nella libreria con lo stesso nome.

Se come operando lib-ref è specificato lib-name, il file da sostituire è cercato nella prima libreria aperta con nome lib-name.

Se come operando lib-ref è specificato (,unit-name), il file da sostituire è cercato nella prima libreria aperta residente sul disco o floppy disk nell'unità con nome simbolico unit-name.

Se l'operando lib-ref non è specificato, il file da sostituire è cercato nella prima libreria aperta, secondo l'ordine di apertura.

Note

1. Non si può utilizzare il comando REPLACE per sostituire un programma o file testo con un programma o file testo appena introdotto da tastiera, quindi senza nome. In questo caso si introduca il comando PURGE per cancellare il file che si vuole sostituire e quindi il comando SAVE per registrare il nuovo file.

2. Eseguendo il comando REPLACE si può verificare una condizione anomala che causa la cancellazione del file originale dalla libreria senza sostituirlo; a questo punto il sistema invia il seguente messaggio:

   ERROR cod. errore FILE filename PURGED

   Il file in memoria principale non viene cancellato e si può procedere all'azione correttiva. Pèr esempio si può eseguire un comando SAVE riferito ad una diversa libreria.

Esempio

Si aggiorni il programma EQUAZ presente nella libreria MAT sul disco nell'unità UD.

Premere [SHIFT] [OLD] [E] [Q] [U] [A] [Z] [,] [(] [M] [A] [T] [,] [U] [D] [)] [END OF LINE]

Il programma è trasferito in memoria principale.

Premere [LIST] [END OF LINE]

E' stampato il listing del programma. Si introducano le modifiche desiderate.

Premere [R] [E] [P] [(] [M] [A] [T] [,] [U] [D] [)] [END OF LINE]

Il nuovo programma EQUAZ è registrato nella libreria sostituendo il vecchio programma con lo stesso nome.

Comando RESEQUENCE

Funzione

Modifica la numerazione delle linee del programma o del file testo presente in memoria principale.

Formato

RES [EQUENCE] [line-num] [, increment]

dove:

line-num
indica il primo numero di linea da assegnare al programma o file testo presente in memoria principale

increment
è un numero intero positivo che indica il valore che è aggiunto ad ogni numero di linea per ottenere il numero di linea successivo

Azione

Il comando ordina al sistema di assegnare alla prima linea del programma o file testo presente in memoria principale il numero specificato con il primo operando e di aggiungere ad ogni numero di linea l'incremento specificato con il secondo operando, per assegnare il successivo numero di linea.

Il comando, con solamente il primo operando, ordina al sistema di assegnare alla prima linea, del programma o file testo, il numero specificato e di aggiungere 10 ad ogni numero di linea per assegnare il numero di linea successivo.

Il comando, con solamente il secondo operando, ordina al sistema di assegnare il numero increment alla prima linea del programma o file testo e di aggiungere ad ogni numero di linea l'incremento specificato con il secondo operando per assegnare il successivo numero di linea.

Il comando, privo di operandi, ordina al sistema di

assegnare il numero 10 alla prima linea del programma o file testo e di aggiungere 10 ad ogni numero di linea per assegnare il successivo numero di linea.

Note

1. Il sistema modifica automaticamente i riferimenti ad altre istruzioni di programma contenuti in alcune istruzioni (es. GOTO etc.) presenti in memoria principale.

2. Se in un programma una istruzione fa riferimento ad altra istruzione che è stata cancellata, il comando RESEQUENCE rinumera i numeri di linea con l'incremento specificato, ma stampando il listing del programma (comando LIST) si potrà notare un gap nella numerazione delle linee. Per rimuovere tale anomalia si esegua il comando DECOMPILE e quindi il comando COMPILE.

3. Il comando non può essere usato con i programmi protetti mediante il comando SECURE.

Esempi

1. Si rinumeri il seguente programma presente in memoria principale iniziando da 10 con incremento di 10.

```
FILE

0040 GOTO 55
0045 PRINT "VUOI ESEGUIRE UN'ALTRA MEDIA?"
0048 INPUT A$
0050 IF A$<>"SI" THEN 500
0055 PRINT "DIMMI DI QUANTI NUMERI VUOI FARE LA MEDIA"
0090 INPUT N
0100 LET X=0
0120 FOR I=1 TO N STEP 1
0150 INPUT M
0170 PRINT M
0200 LET X=M+X
0240 NEXT I
0270 PRINT "LA MEDIA E':"
0300 PRINT X/N
0340 GOTO 45
0500 END

END OF LISTING
```

Premere R E S END OF LINE

Premere L I S END OF LINE

```
FILE

0010 GOTO 50
0020 PRINT "VUOI ESEGUIRE UN'ALTRA MEDIA?"
0030 INPUT A$
0040 IF A$<>"SI" THEN 160
0050 PRINT "DIMMI DI QUANTI NUMERI VUOI FARE LA MEDIA"
0060 INPUT N
0070 LET X=0
0080 FOR I=1 TO N STEP 1
0090 INPUT M
0100 PRINT M
0110 LET X=M+X
0120 NEXT I
0130 PRINT "LA MEDIA E`:"
0140 PRINT X/N
0150 GOTO 20
0160 END

END OF LISTING
```

Come si vede i riferimenti ad altre istruzioni contenuti in alcune di esse sono rinumerati automaticamente.

2. Si rinumeri il seguente programma presente in memoria principale come file testo.

```
FILE

0040 GOTO 55
0045 PRINT "VUOI ESEGUIRE UN'ALTRA MEDIA?"
0048 INPUT A$
0050 IF A$<>"SI" THEN 500
0055 PRINT "DIMMI DI QUANTI NUMERI VUOI FARE LA MEDIA"
0090 INPUT N
0100 LET X=0
0120 FOR I=1 TO N STEP 1
0150 INPUT M
0170 PRINT M
0200 LET X=M+X
0240 NEXT I
0270 PRINT "LA MEDIA E`:"
0300 PRINT X/N
0340 GOTO 45
0500 END

END OF LISTING
```

Premere R E S END OF LINE

Premere L I S END OF LINE

```
FILE

0010 GOTO 55
0020 PRINT "VUOI ESEGUIRE UN'ALTRA MEDIA?"
0030 INPUT A$
0040 IF A$<>"SI" THEN 500
0050 PRINT "DIMMI DI QUANTI NUMERI VUOI FARE LA MEDIA"
0060 INPUT N
0070 LET X=0
0080 FOR I=1 TO N STEP 1
0090 INPUT M
0100 PRINT M
0110 LET X=M+X
0120 NEXT I
0130 PRINT "LA MEDIA E`:"
0140 PRINT X/N
0150 GOTO 45
0160 END

END OF LISTING
```

Come si vede i riferimenti ad altre istruzioni non sono modificati.

Comando RUN

Funzione

Comanda l'esecuzione di un programma.

Formato

RUN [**filename** [, **lib-ref**] / **line-num**]

dove:

filename
è il nome di un programma presente in una libreria

line-num
indica il numero di linea da cui inizia l'esecuzione di un programma presente in memoria principale

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

Azione

Il comando ordina al sistema di caricare in memoria principale ed eseguire il programma di nome filename, presente nella libreria lib-name sul disco o floppy disk nell'unità con nome simbolico unit-name.

Il comando, con line-num come operando, comunica al sistema di eseguire il programma presente in memoria principale iniziando l'esecuzione dall'istruzione il cui numero di linea è line-num.

Il comando, senza operandi, ordina al sistema di eseguire il programma che è presente in memoria principale.

Note

1. Se come operando lib-ref è specificato lib-name, il programma da eseguire è cercato nella prima libreria aperta di nome lib-name.

2. Se come operando lib-ref è specificato (,unit-name), il programma da eseguire è cercato nella prima libreria aperta residente sul disco o floppy disk nell'unità con nome simbolico unit-name.

3. Se l'operando lib-ref non è specificato, il programma da eseguire è cercato nella prima libreria aperta, secondo l'ordine di apertura.

4. Per poter eseguire un programma devono essere effettuati una serie di controlli di coerenza fra le istruzioni del programma, ad esempio se nel programma vi è una istruzione:

   55 GOTO 95

   si deve verificare l'esistenza della istruzione 95. Questa fase (detta preesecuzione) è attuata eseguendo il comando PREPARE (vedi PREPARE) oppure il comando RUN, se il programma non è stato precedentemente registrato in una libreria con l'informazione che è formalmente corretto. Per introdurre tale informazione nel programma si possono seguire due vie:

   - interrompere l'esecuzione del programma premendo il tasto di console BREAK e quindi registrarlo in una libreria con il comando SAVE o REPLACE

   - preeseguire il programma con il comando PREPARE e quindi registrarlo in una libreria con il comando SAVE o REPLACE

   In questo modo i successivi comandi RUN eseguono direttamente il programma evitando la fase di preesecuzione.

5. Se, durante la fase di preesecuzione, il sistema non rileva alcun errore, viene visualizzato il messaggio: *** FORMALLY CORRECT PROGRAM **** che è anche stampato se la funzione PRINT ALL è abilitata. Viene quindi visualizzato il messaggio: PROGRAM nome-programma RUNNING; il programma è eseguito; al termine della esecuzione viene visualizzato il mes-

saggio: READY.

6. Normalmente, durante l'esecuzione di un programma, appare sul display il seguente messaggio: PROGRAM nome-programma RUNNING ed al termine della esecuzione appare sul display il messaggio: READY. Se si vogliono sopprimere tali messaggi, si deve specificare l'operando MSG nel comando SAVE quando si comanda la registrazione del programma in una libreria (vedi comando SAVE).

7. Per poter eseguire un programma con il comando RUN, nella prima libreria aperta vi deve essere spazio disponibile pari allo spazio occupato in memoria principale dal programma stesso. Se però si introduce il comando RUN filename, il programma è eseguito anche se non è verificata la condizione suddetta. Se il programma è stato appena creato, si modifichi l'ordine con cui le librerie sono state aperte.

8. L'esecuzione di un programma può essere terminata premendo il tasto di console BREAK ; il sistema passa nello stato comandi.

9. Si può interrompere l'esecuzione di un programma premendo il tasto di console STEP ; il sistema passa nello stato di debugging. Premendo successivamente STEP si possono eseguire le istruzioni del programma una per volta oppure, premendo CONTINUE si esegue completamente il resto del programma (vedi capitolo 7).

10. Non si può utilizzare come operando line-num un numero di linea di una istruzione interna ad un ciclo FOR/NEXT, ad un sottoprogramma o ad una definizione di funzione multilinea.

Comando SAVE

Funzione

Registra un programma od un file testo, presente in memoria principale, in una libreria.

Formato

SAV[E] filename, [lib-ref] [,MSG=n] [,COM=m]

dove:

filename
indica il nome con cui il programma od il file testo sarà registrato nella libreria

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(, unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

MSG=n
con n uguale a zero od a uno; inibisce la visualizzazione di uno o tutti i messaggi emessi durante l'esecuzione del comando RUN

COM=m
con m numero intero di al massimo 5 cifre, specifica che all'atto del caricamento del programma in memoria principale m byte della memoria saranno riservati come area comune per lo scambio di dati con altri programmi (vedi istruzione COMMON)

Azione

Il comando ordina al sistema di registrare, con il nome filename, nella libreria lib-name sul disco o floppy disk nell'unità con nome simbolico unit-name il programma o file testo presente in memoria principale.

Se è specificato l'operando MSG=Ø, il comando SAVE inibisce la visualizzazione di tutti i messaggi emessi

durante l'esecuzione del comando RUN.

Se è specificato l'operando MSG=1, il comando SAVE inibisce la visualizzazione del messaggio: PROGRAM nome-programma RUNNING, durante l'esecuzione del comando RUN.

Se il comando SAVE agisce su un programma e se è specificato l'operando COM=m, il programma viene registrato con l'indicazione che all'atto del caricamento in memoria principale (comando OLD, oppure RUN, oppure istruzione CHAIN) dovranno essere riservati m byte all'inizio dell'area utente, subito prima dell'indirizzo da cui parte il caricamento del programma. L'area così riservata potrà essere utilizzata dal programma per depositare o prelevare dati comuni ad altri programmi (vedi istruzione COMMON).

Note

1. Se come operando lib-ref è specificato lib-name, il file è registrato nella prima libreria aperta di nome lib-name.

2. Se come operando lib-ref è specificato (,unit-name), il file è registrato nella prima libreria aperta residente sul disco o floppy disk nell'unità con nome simbolico unit-name.

3. Se l'operando lib-ref non è specificato, il file è registrato nella prima libreria aperta, secondo l'ordine di apertura.

4. Gli operandi MSG=n e COM=m possono essere specificati solamente se in memoria principale è presente un programma.

5. La capacità della zona di memoria specificata in m byte è arrotondata per eccesso al più vicino multiplo di quattro.

6. La dimensione specificata per l'area comune può essere modificata eseguendo la seguente sequenza di comandi:

   OLD filename

PURGE filename
SAVE filename, (lib-name, unit-name), COM=m

con m che specifica il nuovo valore.

7. Se il tentativo di caricare in memoria principale un programma registrato con un comando SAVE con l'operando COM=m causa una segnalazione di errore che specifica spazio di memoria utente disponibile insufficiente, si prema il tasto [CLEAR] e si esegua il comando LIST. Se è stampato un listing, il programma è stato caricato in memoria principale ma senza l'area comune richiesta. In questo caso la dimensione dell'area comune può essere ridotta eseguendo un comando PURGE seguito da un comando SAVE che specifica un valore più piccolo di m. Si può quindi ripetere l'operazione che ha provocato il messaggio d'errore.

8. Il comando SAVE non può essere usato per registrare un programma o file testo in una sottolibreria package che sia stata protetta mediante l'esecuzione del programma di utilità LBPROTECT.

9. L'operando filename non può essere uguale al nome di un programma, file testo o file dati presente nella libreria in cui il file testo o programma deve essere registrato.

10. La parola chiave del comando può essere introdotta premendo [SHIFT] con [SAVE].

11. Per registrare in una libreria dei programmi che sono ottimizzati in occupazione di memoria e velocità di esecuzione si devono eseguire i seguenti comandi nell'ordine con cui sono riportati:

    DECOMPILE
    COMPILE
    PREPARE

    Dopo aver eseguito il comando PREPARE si prema il tasto di console [BREAK] e si esegua il comando SAVE.

Esempio

Si registri il programma presente in memoria principale, nella libreria MAT sul disco nell'unità UD, con il nome CAL.

Premere SHIFT SAVE C A L , (M A T , U D) END OF LINE

Comando SECURE

Funzione

Vieta l'esecuzione di alcuni comandi su parte o tutto un programma o file dati.

Formato

SEC[URE] filename, [lib-ref] [,n]

dove:

filename

indica il nome di un file programma o file dati, presente in una libreria che si vuole proteggere da alcune operazioni

lib-ref

può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(, unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

n

è un numero intero positivo che indica la prima linea di un insieme di linee da proteggere (od il primo dato di un insieme di dati da proteggere)

Azione

Se filename è il nome di un programma presente in una libreria, il comando SECURE ordina al sistema di proteggerlo dall'azione dei comandi:

DECOMPILE
LINK
MODIFY
RESEQUENCE

DELETE LINE
FETCH
LIST

e dall'azione dei tasti [↑] e [↓]. Se nel comando è specificato l'operando n, allora l'azione dei comandi:

DELETE LINE
FETCH
LIST

e dei tasti [↑] e [↓], è impedita dall'istruzione con il numero di linea n in poi.

Se filename è il nome di un file dati presente in una libreria, il comando SECURE ordina al sistema di proteggerlo dall'azione dei comandi MODIFY, TRANSCODE e TRUNCATE e di impedire la registrazione di nuovi dati in tutto il file. Il comando SECURE impedisce anche la stampa del contenuto di un file dati effettuata con il programma di utilità FLPRINT; in questo caso, se si specifica l'operando n, tale stampa è impedita dall'ennesimo dato in poi.

Note

1. Se come operando lib-ref è specificato lib-name, il file da proteggere è cercato nella prima libreria aperta di nome lib-name.

2. Se come operando lib-ref è specificato (,unit-name), il file da proteggere è cercato nella prima libreria aperta residente sul disco o floppy disk nella unità con nome simbolico unit-name.

3. Se l'operando lib-ref non è specificato, il file da proteggere è cercato nella prima libreria aperta, secondo l'ordine di apertura.

4. Non si può togliere la protezione assegnata ad un programma o file dati mediante il comando SECURE.

5. Si può ampliare ma non si può ridurre l'area di protezione di un programma. Così, se un programma è stato protetto con il comando:

   SECURE MAT1,,100

   si può ampliare l'area protetta con il comando:

   SECURE MAT1,,50

   ma non si può ridurre l'area protetta; infatti il

comando: SECURE MAT1,,300 non è accettato.

Esempio

Si protegga il programma MAT1, residente nella prima libreria aperta, dall'istruzione 50 in poi.

Premere [S][E][C] [M][A][T][1][.][.][5][0] [END OF LINE]

Comando SHIFT

Funzione

Modifica la numerazione delle linee di un programma o file testo presente in memoria principale, iniziando dalla linea specificata.

Formato

SHI [FT] line-num, increment

dove:

line-num

è un numero di linea che specifica da quale linea deve essere modificata la numerazione delle linee

increment

è un numero intero positivo che specifica il valore di cui deve essere incrementato ogni numero di linea da modificare

Azione

Il comando ordina al sistema di modificare i numeri di linea delle linee del programma, o file testo, presente in memoria principale, iniziando dalla linea con numero di linea line-num. I nuovi numeri di linea sono ottenuti aggiungendo il valore specificato con increment ai precedenti numeri di linea.

Note

1. Se in memoria principale vi è un programma, l'esecuzione del comando SHIFT modifica automaticamente i riferimenti ad altre istruzioni di programma contenuti in alcune istruzioni (es. GOTO). Sono modificati anche i riferimenti ad istruzioni che non sono state ancora introdotte nel programma o che sono state cancellate.

2. Se si cancellano delle linee di programma che contengono un riferimento ad altre istruzioni, il riferimento è ricordato dal sistema e quando si esegue il comando SHIFT, viene incrementato di increment. Per rimuovere tali riferimenti si devono eseguire i comandi DECOMPILE e COMPILE, nell'ordine.

Esempio

Vediamo come i numeri di linea, dal 15 in poi, sono incrementati di 100 nella routine sottostante.

```
LIS
FILE

0010 INPUT A$
0015 PRINT A$
0040 INPUT B$
0070 PRINT B$
0080 END

END OF LISTING


SHI 15,100
LIS
FILE

0010 INPUT A$
0115 PRINT A$
0140 INPUT B$
0170 PRINT B$
0180 END

END OF LISTING
```

Comando SPACE

Funzione

Visualizza lo spazio disponibile (in byte) per registrazioni successive in una libreria.

Formato

SPA[CE] [lib-ref]

dove:
lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di unità a disco od a floppy disk

Azione

Il comando ordina al sistema di visualizzare lo spazio libero da registrazioni nella libreria di nome lib-name residente sul disco o floppy disk nell'unità con nome simbolico unit-name.

Note

1. Se come operando lib-ref è specificato lib-name, la libreria di cui è visualizzato lo spazio libero disponibile è la prima libreria di nome lib-name aperta.

2. Se come operando lib-ref è specificato (,unit-name), la libreria di cui è visualizzato lo spazio libero disponibile è la prima libreria aperta residente nell'unità con nome simbolico unit-name.

3. Se l'operando lib-ref non è specificato, la libreria di cui è visualizzato lo spazio libero disponibile è la prima libreria aperta.

Esempio

Si richieda la visualizzazione dello spazio libero nella libreria MAT presente sull'unità floppy disk FDU1.

Premere [S] [P] [A] [(] [M] [A] [T] [.] [F] [D] [U] [1] [)] [END OF LINE]

Comando START

Funzione

Permette di riprendere l'esecuzione di un programma da una istruzione specificata.

Formato

STA [RT] line-num

dove:
line-num
è un numero intero positivo che indica l'istruzione da cui riprenderà l'esecuzione del programma.

Azione

Il comando ordina al sistema di riprendere l'esecuzione del programma interrotto dalla istruzione con numero di linea line-num.

Note

1. Il comando può essere usato solamente quando il sistema è nello stato di debugging.

2. L'esecuzione del programma riprende immediatamente dopo l'esecuzione del comando START.

3. Non si può avere come operando line-num un numero di linea di una istruzione interna ad un ciclo FOR/NEXT, ad un sottoprogramma o ad una definizione di funzione multilinea.

Esempi

Vedi il capitolo 7.

Comando STKEYS

Funzione	Registra su disco sistema o floppy disk sistema il contenuto dei tasti funzione.
Formato	**STK [EYS]**
Azione	Il comando ordina al sistema di registrare sul disco sistema o floppy disk sistema l'attuale contenuto dei tasti funzione.
Nota	Ogni qual volta il sistema è inizializzato (dopo la accensione, eseguendo il comando CONFIGURE od il comando OPTIONS) od è eseguito il comando LDKEYS, ai tasti funzione è assegnato il contenuto registrato sul disco sistema o floppy disk sistema dall'ultimo comando STKEYS eseguito.

Comando STOP

Funzione

Interrompe l'esecuzione di un programma prima della istruzione specificata.

Formato

STO [P] line-num

dove:
line-num
è un numero intero positivo che indica l'istruzione di programma prima della quale l'esecuzione deve essere interrotta

Azione

Il comando ordina al sistema di interrompere l'esecuzione del programma prima che l'istruzione con numero di linea line-num sia eseguita.

Note

1. Il comando può essere introdotto solamente quando il sistema è nello stato di debugging.

2. Dopo aver introdotto il comando STOP si può riprendere l'esecuzione del programma premendo il tasto di console CONTINUE oppure STEP , o introducendo il comando START.

3. Il comando STOP non deve essere confuso con l'istruzione STOP (vedi capitolo 5).

Esempi

Vedi il capitolo 7.

Comando TEXT

Funzione

Permette di introdurre un file testo da tastiera.

Formato

TEX [T]

Azione

Il comando comunica al sistema che quanto verrà successivamente introdotto sono linee di un file testo.

Note

1. Quando il comando è eseguito il precedente contenuto della memoria utente è cancellato.

2. Utilizzando il comando AUTO# si possono numerare automaticamente le linee del file testo introdotte successivamente.

3. Per la definizione di file testo si veda il capitolo 2 e per ulteriori informazioni sull'impiego dei file testo si veda la pubblicazione P6066 - Gestione dei file testo - Guida dell'utente.

Comando TRANSCODE

Funzione

Converte un file dati sequenziale in un file testo e viceversa.

Formato

TRA[NSCODE] $\left\{\begin{matrix}T\\D\end{matrix}\right\}$**, filename, [lib-ref] [,#]**

dove:

T

indica che il file dati sequenziale filename presente in una libreria deve essere convertito in un file testo residente in memoria principale

D

indica che il file testo presente in memoria principale deve essere convertito in un file dati residente in una libreria

filename

specifica il nome di un file dati sequenziale

lib-ref

può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con libe-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

#

specifica il tipo di numerazione da assegnare al file convertito

Azione

Il comando, con T come primo operando e con l'operando # , ordina al sistema di convertire ogni dato di un file dati, i cui elementi sono stringhe di caratteri, in una linea di file testo assegnando alla linea il numero di linea specificato nel dato stesso. Se nel comando non è specificato l'operando # , i numeri di linea sono assegnati automaticamente con incremento

di 1. Il file testo prodotto è presente in memoria principale.

Il comando, con D come primo operando, ordina al sistema di creare, nella libreria di nome lib-name residente sul disco o floppy disk nell'unità con nome simbolico unit-name, un file dati di tipo sequenziale con il nome filename e di allocare per esso n byte (dove n è il primo multiplo di 256 nel caso di disco, o 128 nel caso di floppy disk, più grande del numero di byte occupati dal file testo presente in memoria principale). Ogni linea del file testo presente in memoria principale viene quindi convertita in una stringa di caratteri che è registrata nel file dati creato. Se nel comando è specificato l'operando # , il numero di linea di ogni linea del testo è compreso nella stringa registrata come elemento del file dati suddetto. Se l'operando # non è specificato, il numero di linea viene omesso nella conversione delle linee del file testo nel corrispondente elemento del file dati suddetto.

Note

1. Il comando TRANSCODE non può convertire in un file testo un file dati che è stato protetto mediante il comando SECURE.

2. Se nel file dati filename non sono presenti i numeri di linea, il comando TRANSCODE con l'operando # non è accettato ed è visualizzato un messaggio d'errore. Mentre, se il comando TRANSCODE è specificato senza l'operando # , il file dati è trascodificato in un file testo numerato automaticamente dal sistema iniziando da 1 con incremento di 1.

3. Non si può trascodificare un file testo in un file dati che deve essere registrato in una sottolibreria package protetta mediante l'esecuzione del programma di utilità LBPROTECT.

4. Se come operando lib-ref è specificato lib-name, la libreria interessata dal comando TRANSCODE è la prima libreria aperta di nome lib-name.

5. Se come operando lib-ref è specificato (,unit-name), la libreria interessata dal comando TRANSCODE è la prima libreria aperta residente sul disco o floppy disk nell'unità con nome simbolico unit-name.

6. Se l'operando lib-ref non è specificato, la libreria interessata dal comando TRANSCODE è la prima libreria aperta, secondo l'ordine di apertura.

7. Se durante la traduzione di un file dati in un file testo viene emesso un messaggio che avverte l'utente che non vi è spazio sufficiente in memoria utente per continuare l'operazione, le stringhe del file dati che sono state tradotte fino a quel momento sono presenti in memoria principale.

Esempi

1. Si converta un file testo presente in memoria principale in un file dati che sia registrato nella prima libreria aperta con il nome MAT2.

   Premere [T][R][A] [D][.][M][A][T][2][.][#][END OF LINE]

2. Si converta il file dati presente nella prima libreria aperta con il nome MAT2 in un file testo con i numeri di linea contenuti negli elementi del file dati.

   Premere [T][R][A] [T][.][M][A][T][2][.][#][END OF LINE]

Comando TRUNCATE

Funzione

Modifica la lunghezza di allocazione di un file dati sequenziale eguagliandola alla sua lunghezza attuale.

Formato

TRU[NCATE] filename [,lib-ref]

dove:
filename
indica il nome di un file dati di tipo sequenziale
lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

Azione

Il comando comunica al sistema di eguagliare la lunghezza di allocazione del file dati, specificato con l'operando filename, alla sua lunghezza attuale.

Note

1. Lo spazio non effettivamente occupato dal file dati è reso disponibile per altre registrazioni nella medesima libreria.

2. Se la lunghezza attuale del file dati specificato non è multiplo di 256, nel caso di disco, o 128 nel caso di floppy disk, allora lo spazio allocato dal sistema è uguale al numero di byte pari al multiplo di 256, o di 128, successivo al valore che indica la lunghezza attuale.

3. Se come operando lib-ref è specificato lib-name, il file è cercato nella prima libreria aperta di nome lib-name.

4. Se come operando lib-ref è specificato (,unit-name), il file è cercato nella prima libreria aperta residente sul disco nell'unità con nome simbolico unit-name.

5. Se l'operando lib-ref non è specificato, il file è cercato nella prima libreria aperta, secondo l'ordine di apertura.

6. Il comando TRUNCATE non può essere usato con file dati protetti mediante il comando SECURE.

Esempio

Si eguagli la lunghezza di allocazione del file dati sequenziale MAT3, residente nella prima libreria aperta, alla sua lunghezza attuale.

Premere T R U M A T 3 END OF LINE

Comando VALIDATE

Funzione

Rende nuovamente operabile un file dati rimasto aperto a seguito della terminazione anormale di un programma.

Formato

VAL [IDATE] filename [, lib-ref]

dove:

filename
specifica il nome del file da rendere nuovamente operabile

lib-ref
può essere uno dei seguenti operandi:
(lib-name, unit-name)
lib-name
(,unit-name)
con lib-name che specifica il nome di una libreria e unit-name che specifica il nome di una unità a disco od a floppy disk

Azione

Il file dati di nome filename, che è rimasto aperto in seguito alla terminazione in modo anomalo della esecuzione di un programma che predispone il file ad operazione di registrazione od aggiornamento, viene reso nuovamente operabile.

Note

1. Se filename è il nome di un file dati ad accesso sequenziale aperto in registrazione, esso è reso nuovamente disponibile per la registrazione. Se filename è il nome di un file dati ad accesso diretto che era rimasto aperto in aggiornamento, il file è reso nuovamente disponibile in lettura e registrazione. In ogni caso, se per l'effetto della terminazione anormale, alcune informazioni erano state danneggiate, queste non sono recuperabili.

2. Per determinare se un file è aperto si esegua il comando CATALOG.

3. Se come operando lib-ref è specificato lib-name, il file è cercato nella prima libreria aperta di nome lib-name.

4. Se come operando lib-ref è specificato (,unit-name), il file è cercato nella prima libreria aperta residente sul disco o floppy disk nell'unità con nome simbolico unit-name.

5. Se l'operando lib-ref non è specificato, il file è cercato nella prima libreria aperta.

4. BASIC: DATI, VARIABILI, ESPRESSIONI E FILE DATI

Questo capitolo offre al lettore che ha già una certa familiarità con i concetti del linguaggio BASIC la possibilità di effettuare una rapida e completa consultazione di tutte le possibilità offerte dal linguaggio BASIC realizzato per il sistema P6066. Nel capitolo successivo sono descritte tutte le istruzioni del linguaggio BASIC.

I caratteri del linguaggio BASIC

I caratteri che hanno un ruolo sintattico nel linguaggio BASIC sono classificabili in tre categorie:

- caratteri alfabetici
- caratteri numerici
- caratteri speciali

Tutti gli elementi che costituiscono un programma BASIC sono composti con caratteri che appartengono ad una delle suddette categorie, meno le istruzioni di commento (REM) e le costanti stringa che possono essere composte con qualsiasi carattere che si può introdurre da tastiera (vedi set di caratteri P6066 nell'appendice E).

Caratteri alfabetici

I caratteri alfabetici sono tutte le lettere maiuscole dell'alfabeto inglese.

Caratteri numerici

I caratteri numerici sono le cifre del sistema decimale da Ø a 9.

Caratteri speciali

I caratteri speciali sono indicati nella tabella 4-1.

Carattere	Nome
	Spazio
=	Uguale o simbolo di assegnazione
+	Segno più
-	Segno meno
*	Asterisco o segno di moltiplicazione
/	Divisione
↑	Elevamento a potenza
(	Parentesi aperta
)	Parentesi chiusa
,	Virgola
"	Apice
;	Punto e virgola
.	Fine periodo e punto decimale
:	Due punti
>	Maggiore di
<	Minore di
#	Simbolo di numero
$	Simbolo di dollaro

Tabella 4-1 Caratteri speciali

Alcuni caratteri speciali possono essere combinati per comporre i seguenti elementi che hanno un ruolo sintattico nel linguaggio BASIC:

>= maggiore o uguale a
=> maggiore o uguale a
<= minore o uguale a
=< minore o uguale a
<> non uguale a
>< non uguale a

Gli spazi

Gli spazi non hanno un significato sintattico meno che nelle costanti stringa e nella istruzione immagine (vedi capitolo 5) che specifica il formato dei dati da stampare, visualizzare sul display o trasferire ad una unità periferica.

Gli spazi, tuttavia, non sono ammessi all'interno di:

- un numero di linea
- una parola chiave
- un nome di variabile o di funzione

- una costante numerica
- un operatore di confronto.

Dati numerici

Nel linguaggio BASIC i dati numerici sono dati con un valore numerico espresso nel sistema decimale.

Grandezza di un numero

Si definisce grandezza di un numero il suo valore assoluto. Il BASIC P6066 permette di trattare numeri la cui grandezza sia superiore od uguale a 10^{-99} e minore di od uguale a $9.999999999999*10^{99}$.

Precisione

Il linguaggio BASIC permette di utilizzare due tipi di precisione per le variabili numeriche: singola e doppia. Il grado di precisione per le due forme è rispettivamente di 6 e 13 cifre significative. Si può scegliere il tipo di precisione con cui devono essere rappresentati i valori delle variabili di un programma impiegando l'istruzione DCL (vedi capitolo 5).

Nella rappresentazione in singola precisione il valore di una variabile è rappresentato, in memoria principale, con un numero nella forma $M*10^N$. M è un numero che assume la forma X.YYYYY (con $1 \leq X \leq 9$ e $0 \leq Y \leq 9$), mentre N è un numero intero compreso tra -63 e +63. Il valore di una variabile numerica in singola precisione occupa 8 byte in memoria principale. Nella figura 4-1 è rappresentato schematicamente il campo dei valori numerici che possono assumere le variabili numeriche in singola precisione.

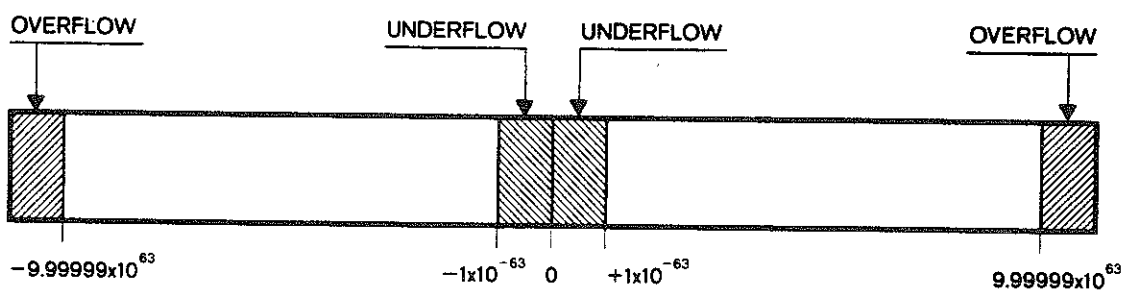


Figura 4-1 Campo di rappresentazione interna dei valori assegnati a variabili in singola precisione

I valori minori di $-9.99999*10^{63}$ e maggiori di $9.99999*10^{63}$, sono nella zona di OVERFLOW: essi esprimono valori più grandi in valore assoluto di quelli rappresentabili in memoria principale in singola precisione. I valori maggiori di $\mp 1*10^{-63}$ e minori di $1*10^{-63}$ (escluso lo zero), sono nella zona di UNDERFLOW: essi esprimono valori minori di quelli rappresentabili in memoria principale in singola precisione.

Nella rappresentazione in doppia precisione il valore di una variabile è rappresentato, in memoria principale, con un numero nella forma $R*10^{E}$. R è un numero che assume la forma X.YYYYYYYYYYY ($1 \leq X \leq 9$ e $0 \leq Y \leq 9$), mentre E è un numero intero compreso tra -99 e +99. Il valore di una variabile numerica in doppia precisione occupa 8 byte in memoria principale. Le espressioni sono calcolate sempre in doppia precisione, se almeno uno degli operandi specificati è espresso in doppia precisione, ed i risultati sono poi arrotondati al tipo di precisione scelto per la variabile a cui sono assegnati. I valori delle variabili numeriche sono assunti in doppia precisione se non vi è una dichiarazione contraria nell'ambito del programma. Nella figura 4-2 è rappresentato schematicamente il campo di rappresentazione dei valori numerici che possono assumere le variabili numeriche in doppia precisione.

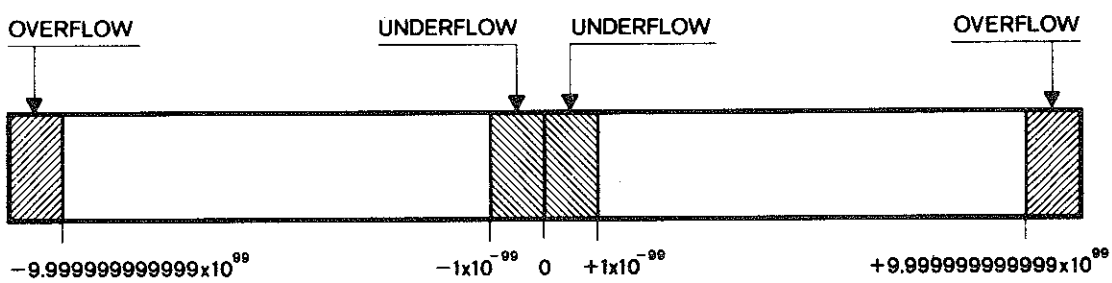


Figura 4-2 Campo di rappresentazione dei valori numerici assegnati a variabili in doppia precisione

I valori minori di $-9.999999999999*10^{99}$ e maggiori di $9.999999999999*10^{99}$ sono nella zona di OVERFLOW: essi esprimono valori più grandi in valore assoluto di quelli rappresentabili in memoria principale in doppia

precisione.

I valori maggiori di $-1*10^{-99}$ e minori di $1*10^{-99}$ (escluso lo zero), sono nella zona di UNDERFLOW: essi esprimono valori minori di quelli rappresentabili in memoria principale in doppia precisione.

Formato dei dati numerici

I dati numerici possono essere introdotti, visualizzati o stampati come:

- interi
- decimali in virgola fissa
- decimali in virgola mobile

La scelta del formato dipende dalla grandezza del numero e dalla precisione richiesta. I numeri possono essere positivi o negativi. I numeri negativi devono essere preceduti dal segno meno. I numeri positivi possono essere preceduti dal segno più.

Numeri interi: I numeri interi sono rappresentati con al massimo 13 cifre significative, precedute dal segno algebrico. Esempi di numeri interi sono:

0 4 +4 -4
9999999999999 (massimo numero intero)
-9999999999999 (minimo numero intero)

Decimali in virgola fissa: I numeri espressi in virgola fissa sono costituiti, al massimo, da 13 cifre significative, precedute opzionalmente dal segno, con interposto tra di esse il punto decimale per distinguere le cifre intere da quelle decimali. Esempi di numeri nel formato in virgola fissa sono:

-.86 .7 5. +2.3
9999999999999. (massimo valore assoluto esprimibile in virgola fissa)

Decimali in virgola mobile: I numeri espressi in virgola mobile (detta anche rappresentazione scientifica), sono rappresentati con un segno (opzionale se il numero è positivo) seguito da un numero intero od in virgola fissa, seguito dal carattere E. Dopo il carattere E segue un altro numero di due cifre preceduto, eventualmente, dal segno. Il valore di un numero rappresentato in virgola mobile è uguale al numero che

precede il simbolo E (mantissa) moltiplicato per 10, elevato alla potenza con esponente indicato dal numero che segue il simbolo E (esponente). Si osservi che il simbolo E deve essere sempre preceduto da un numero, quindi 10^8 deve essere introdotto come: 1E8. Esempi di numeri nel formato in virgola mobile sono: .99E-5 +1E5 2.E3 -3.0E+1 i cui valori sono rispettivamente: .0000099 100000 2000 -30

Costanti numeriche

Una costante numerica è un numero intero, un numero decimale in virgola fissa od un numero decimale in virgola mobile, il cui valore non viene mai modificato durante l'esecuzione di un programma. Negli esempi che seguono 94, 9.4 e 9.4E-10, sono costanti numeriche:

```
150 LET A=A+94
160 DATA 9.4 , 9.4E-10
```

La costante numerica π è predefinita dal sistema; il suo valore è: 3.141592653590. Per utilizzare π in un programma la si richiama mediante il nome PI, come nell'esempio che segue

```
200 LET A=PI*R*R
```

Una costante numerica occupa un numero di byte, in memoria principale, variabile in funzione del numero di cifre di cui si compone secondo la seguente ella:

Cifre	Byte occupati
1	5
2	6
3	6
4	7
5	7
6	8
7	8
8	9
9	9
10	10
11	10
12	11
13	11

così ad esempio:

123.21	occupa 7 byte
-.83913E-23	occupa 7 byte
1234.5678E12	occupa 9 byte
31	occupa 6 byte
1	occupa 5 byte
1.0E2	occupa 6 byte

Se in un programma la stessa costante è riferita più di una volta si può risparmiare spazio di memoria assegnandola ad una variabile numerica e quindi utilizzando quest'ultima al suo posto.

Variabili semplici numeriche

Una variabile semplice numerica è un dato numerico, richiamato in una istruzione di programma mediante un nome, il cui valore può essere modificato durante l'esecuzione del programma. Il nome associato ad una variabile semplice numerica è costituito da una lettera maiuscola dell'alfabeto inglese o da una lettera seguita da una cifra decimale (da Ø a 9).

Sono esempi di variabili semplici numeriche: Z B6 M2. Ad una variabile semplice numerica può essere assegnato un valore da tastiera (vedi istruzione INPUT) o da programma (vedi istruzioni READ, DATA o LET). In un programma si possono utilizzare fino a 123 variabili semplici numeriche.

Quando s'inizia l'esecuzione di un programma, tutte le variabili numeriche sono inizializzate con un valore "non definito". Se ad una variabile numerica non è stato assegnato un valore, nel momento in cui essa è utilizzata in una istruzione di programma, il sistema le assegna il valore zero ed esegue l'istruzione. Eseguita l'istruzione il sistema è nello stato di debugging e viene visualizzato un messaggio di errore di tipo recuperabile (vedi appendice D). Si può continuare l'esecuzione del programma, premendo il tasto CONTINUE, oppure richiederne la terminazione, premendo il tasto BREAK.

Una variabile semplice numerica è rappresentata in memoria principale in doppia precisione, a meno che non si sia precisata, con l'istruzione DCL, la semplice precisione; in questo caso l'elaborazione è più rapida. Per assegnare il risultato della esecuzione di un'espressione numerica ad una variabile numerica, si

utilizza l'istruzione LET. Se il risultato della suddetta espressione è un valore compreso in una zona di OVERFLOW (vedi paragrafo "Precisione"), alla variabile è assegnato il valore $-9.999999999999*10^{99}$ oppure $9.999999999999*10^{99}$ ($-9.99999*10^{63}$ oppure $9.99999*10^{63}$, se la variabile è stata dichiarata in singola precisione). Se il risultato della suddetta espressione è un valore compreso in una zona di UNDERFLOW, alla variabile è assegnato il valore zero. Sia nel caso di OVERFLOW che nel caso di UNDERFLOW l'esecuzione del programma è interrotta, il sistema commuta nello stato di debugging, e sul display appare un messaggio di errore di tipo recuperabile (vedi appendice D). L'operatore può scegliere di proseguire nell'esecuzione del programma, premendo il tasto di console CONTINUE, o di terminare l'esecuzione del programma, premendo il tasto di console BREAK.

Nota: Una variabile semplice numerica sia in singola che in doppia precisione occupa dieci byte in memoria principale. In entrambi i casi due byte sono occupati dal nome della variabile ed il resto dal valore numerico. Nel caso in cui le variabili semplici siano dichiarate in singola precisione (vedi istruzione DCL) le relative elaborazioni sono eseguite più rapidamente che nel caso in cui esse siano in doppia precisione.

Dati non numerici

Si definiscono dati non numerici tutti quei dati a cui non corrisponde un valore numerico. I dati non numerici si distinguono in:

- costanti stringa
- variabili stringa.

Costanti stringa

Una costante stringa è costituita da una qualunque sequenza di caratteri del set P6066 (vedi appendice E) racchiusa tra virgolette. Il carattere virgolette non può far parte di una costante stringa. In alcuni casi si può definire una costante stringa senza utilizzare le virgolette come inizio e fine della stringa (vedi l'istruzione DATA nel capitolo 5). Esempi di costanti sono:

```
" GRANDEZZA: Temperatura "
"*** FINE DELLA ROUTINE MATRICI***"
" 1815 - 1945 "
```

Si definisce lunghezza di una costante stringa il numero di caratteri che la compongono, inclusi gli spazi ed escluse le virgolette (la cui unica funzione è quella di delimitare l'inizio e la fine della stringa). La massima lunghezza di una costante stringa è di 75 caratteri. Una costante stringa occupa in memoria principale tanti byte quanti sono i caratteri che la compongono, più due byte.

Variabili semplici stringa

Una variabile semplice stringa è costituita da una sequenza di caratteri del set P6066 che può essere modificata durante l'esecuzione di un programma. Le istruzioni di un programma fanno riferimento ad una variabile stringa mediante un nome che è composto da una lettera maiuscola dell'alfabeto (da A a Z) seguita dal simbolo $ oppure da una lettera maiuscola seguita da una cifra decimale (da Ø a 9) e dal simbolo $. Alcuni esempi di nomi di variabili stringa sono:

Z$ B6$ GØ$.

In un programma si possono utilizzare fino a 256 variabili stringa. Il valore può essere assegnato ad una variabile stringa da tastiera (vedi istruzioni INPUT ed RKB) o da programma (vedi istruzioni READ, DATA e LET).

Quando inizia l'esecuzione di un programma, tutte le variabili stringa sono inizializzate con un valore "non definito". Se, quando una variabile stringa è utilizzata in una istruzione di programma, non è stato assegnato ad essa un valore, il sistema le assegna il valore "stringa nulla". La "stringa nulla" è una stringa priva di caratteri; in questo caso, quindi, al nome della variabile suddetta non è associato alcun carattere. L'istruzione viene eseguita ed il sistema commuta nello stato di debugging visualizzando sul display un errore di tipo recuperabile (vedi appendice D). Si può proseguire nella esecuzione del programma, premendo il tasto di console CONTINUE, oppure terminare l'esecuzione, premendo il tasto di console BREAK.

Il massimo numero di caratteri che può essere assegnato ad una variabile stringa (detto lunghezza di allocazione) è specificato mediante una istruzione DCL. Se non viene fatta alcuna dichiarazione, la variabile assume una lunghezza di allocazione di 16 caratteri.

La massima lunghezza di allocazione dichiarabile per una variabile stringa è di 1023 caratteri. Il numero di caratteri che compongono i valori di una variabile stringa è detto lunghezza attuale della variabile stringa. Quando viene assegnata una stringa ad una variabile stringa e le rispettive lunghezze differiscono:

1. Se la stringa ha meno caratteri della lunghezza di allocazione della variabile, quest'ultima assume una lunghezza attuale uguale alla lunghezza della stringa (quindi se, ad esempio, si visualizza il contenuto della variabile, sul display appare solo la stringa assegnata).

2. Se la stringa ha più caratteri della lunghezza di allocazione della variabile, la stringa viene troncata nei caratteri eccedenti tale lunghezza sulla destra e quindi assegnata alla variabile (in questo caso il sistema commuta nello stato di debugging e sul display appare una segnalazione di errore di tipo recuperabile; l'operatore può scegliere di proseguire nell'esecuzione del programma, premendo il tasto di console CONTINUE, oppure di terminare la esecuzione, premendo il tasto di console BREAK).

Nota: Una variabile stringa occupa in memoria principale nove byte, più tanti byte quanti sono stati dichiarati esplicitamente od implicitamente.

Variabili multiple

Si definisce variabile multipla un insieme omogeneo di elementi ognuno dei quali è una variabile numerica od una variabile stringa (i due tipi di variabili non possono coesistere in una variabile multipla).

Una variabile multipla può aver una o due dimensioni. Una variabile multipla ad una dimensione (detta vettore) può essere pensata come una successione naturale di elementi. Una variabile multipla a due dimensioni può essere pensata come una matrice con righe e colonne.

Ad ogni variabile multipla è assegnato un nome. Le regole per l'assegnazione del nome dipendono dal tipo di variabile multipla (vedi paragrafi "Variabili multiple numeriche" e "Variabili multiple stringa"). Agli elementi di una variabile multipla ci si riferisce con lo stesso nome della variabile multipla con in più un

indice (se la variabile è ad una dimensione) o due indici (se la variabile è a due dimensioni) che ne indicano la posizione nell'ambito della variabile suddetta.

Se A è il nome di un vettore i suoi elementi saranno: A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9),A(10). Quindi A(4) è il quarto elemento del vettore A. Se Z è il nome di una matrice i suoi elementi saranno:

Z(1,1)	Z(1,2)	Z(1,3)	Z(1,4)
Z(2,1)	Z(2,2)	Z(2,3)	Z(2,4)
Z(3,1)	Z(3,2)	Z(3,3)	Z(3,4)

Quindi Z(2,3) è l'elemento contenuto nella seconda riga e terza colonna della matrice Z. Gli indici possono essere una qualunque espressione numerica il cui valore, arrotondato all'intero più prossimo, deve essere maggiore di zero e minore o uguale alla corrispondente dimensione che è dichiarata come descritto nel paragrafo successivo.

Dichiarazione delle variabili multiple

Dichiarare una variabile multipla significa specificare:

- il numero di dimensioni (una o due)
- il tipo delle variabili che la compongono (numeriche o stringa)
- il numero di variabili componenti per ogni dimensione
- l'occupazione di memoria principale delle variabili componenti.

Le suddette specificazioni possono essere esplicite od implicite ed esse danno di conseguenza luogo ad altrettante dichiarazioni implicite od esplicite. Le <u>dichiarazioni esplicite</u> sono effettuate mediante le istruzioni DIM e DCL, così le istruzioni:

```
10 DIM A(5,7),B(50,70),C(20),A$(25,24),B$(15)
20 DCL S(A(),B()), 25 A$() , 30 B$()
```

dichiarano esplicitamente che:

A e B sono variabili multiple numeriche a due dimensioni (matrici);
C è una variabile multipla numerica ad una dimensione (vettore);
A$ è una variabile multipla a due dimensioni di tipo stringa;
B$ è una variabile multipla ad una dimensione di tipo stringa.

A può avere al massimo 35 componenti (variabili con indice); in esse il primo indice può variare da 1 a 5 ed il secondo da 1 a 7.

B può avere al massimo 3500 componenti (variabili con indice); in esse il primo indice può variare da 1 a 50 ed il secondo da 1 a 70.

C può avere al massimo 20 componenti (variabili con indice da C(1) a C(20)).

A$ può avere al massimo 600 componenti (variabili con indice); in esse il primo indice può variare da 1 a 25 ed il secondo da 1 a 24.

B$ può avere al massimo 15 componenti (variabili con indice da B$(1) a B$(15)).

A e B hanno come componenti variabili numeriche in singola precisione.

Tutte le variabili componenti di A$ possono avere al massimo 25 caratteri.

Tutte le variabili componenti di B$ possono avere al massimo 30 caratteri.

Nota: Per ulteriori informazioni sulle dichiarazioni esplicite dei diversi attributi delle variabili di un programma si vedano le istruzioni DCL e DIM (capitolo 5).

Le dichiarazioni implicite sono effettuate come conseguenza dell'assenza di istruzioni DIM e DCL riferite alle variabili multiple. Ad esempio in un programma in cui compaiono le istruzioni:

```
90 MAT INPUT A$
100 LET Z(5) = 99
120 LET Y(5,7) = -990.5
250 MAT INPUT A
260 LET B$(5) = "DEVIAZIONE MEDIA"
```

ma non compaiono istruzioni DIM e DCL riferite alle variabili A, A$, B$, Z ed Y, le istruzioni suddette dichiarano implicitamente che:

Z è una variabile multipla numerica ad una dimensione composta da 10 variabili con indice numeriche (da Z(1) a Z(10)), ognuna rappresentata in doppia precisione in memoria principale;

A$ è una variabile multipla stringa a due dimensioni, composta da 25 variabili con indice stringa; ognuna può avere fino a 16 caratteri ed il primo e secondo indice possono variare da 1 a 5;

B$ è una variabile multipla stringa ad una dimensione, composta da 10 variabili con indice stringa (da B$(1) a B$(10)); ognuna può avere fino a 16 caratteri;

Y ed A sono variabili multiple numeriche a due dimensioni composte da 100 variabili numeriche; ognuna è rappresentata in doppia precisione in memoria principale ed il primo e secondo indice possono variare da 1 a 10.

Nota: Ulteriori informazioni sulle dichiarazioni implicite degli attributi delle variabili di un programma sono contenute nella descrizione delle istruzioni DCL e DIM (vedi capitolo 5).

Attenzione: Le dichiarazioni esplicite prevalgono su quelle implicite. L'ultima dichiarazione esplicita (in ordine di numero di linea) prevale sulle precedenti riferite alle stesse variabili multiple.

Ridimensionamento delle variabili multiple

Le dimensioni delle variabili multiple possono essere modificate da alcune istruzioni di programma (per l'elenco completo vedi le istruzioni di tipo MAT nel capitolo 5). Ad esempio nel seguente programma:

```
10 DIM A(6,7), A$(25,10)
-  -  -
90 MAT INPUT B(15,2)
100 MAT INPUT A(2,9)
110 MAT READ: 1,A$(6,5)
-  -  -
-  -  -
9999 END
```

Dopo aver dichiarato, per la matrice A, 42 variabili con indice numeriche (6*7), e per la matrice B 100 variabili con indice numeriche (dichiarazione implicita 10*10), si modificano tali numeri con le istruzioni 90 e 100: infatti A assume 18 variabili con indice numeriche disposte su 2 righe e 9 colonne e B assume 30 variabili con indice numeriche disposte su 15 righe e 2 colonne.

Dopo aver dichiarato, per la variabile multipla a due dimensioni A$, 250 variabili con indice stringa (25*10), l'istruzione 110 modifica tale numero in 30 (6*5).

Infatti per le variabili multiple si hanno dimensioni di allocazione e dimensioni attuali. Le dimensioni di allocazione sono quelle dichiarate esplicitamente (istruzione DIM) od implicitamente (10,10*10 e 5*5) ed esprimono il massimo numero di componenti delle variabili multiple. Le dimensioni attuali si riferiscono al numero di componenti che sono effettivamente utilizzate dalle istruzioni di programma (tale numero è sempre minore od uguale al precedente).

Variabili multiple numeriche

Una variabile multipla numerica contiene come elementi delle variabili con indice numeriche; può avere una dimensione (vettore) o due dimensioni (matrice). Le istruzioni di programma fanno riferimento ad una variabile multipla numerica mediante il suo nome che è costituito da una lettera maiuscola dell'alfabeto inglese. Così A può essere il nome di una variabile multipla numerica mentre A2 non può esserlo. In un programma si può assegnare lo stesso nome ad una variabile semplice numerica e ad una variabile multipla numerica: così B può indicare contemporaneamente il nome di una variabile semplice numerica ed il nome di una variabile multipla numerica. In un programma non si può, tuttavia, assegnare lo stesso nome ad un vettore e ad una matrice. In un programma si possono u-

tilizzare fino a 26 variabili multiple numeriche.
Una variabile multipla numerica può essere ridimensionata.

Le singole componenti di una variabile multipla numerica ad una dimensione (vettore), sono richiamate nelle singole istruzioni di un programma mediante il nome della variabile multipla ed un indice numerico (per questo sono dette variabili con indice) che ne indica la posizione nell'ambito di essa.

Le singole componenti di una variabile multipla a due dimensioni (matrice) sono richiamate nelle singole istruzioni di programma mediante il nome della variabile multipla e due indici numerici (per questo anche esse sono dette variabili con indice) che ne indicano la posizione nell'ambito di essa. Ad esempio nel seguente programma:

```
10 DIM A(50), B(25,25)
-  -  -
-  -  -
-  -  -
100 LET A(50) = 100
110 LET B(10,10) = 1000
-  -  -
-  -  -
-  -  -
9999 END
```

Dopo aver dichiarato 50 componenti per il vettore A e 625 componenti per la matrice B, l'istruzione 100 assegna il valore 100 al cinquantesimo componente del vettore A, mentre l'istruzione 110 assegna il valore 1000 al decimo componente della decima riga della matrice B.

Quando inizia l'esecuzione di un programma tutte le variabili numeriche con indice sono inizializzate con un valore "non definito" e valgono le stesse considerazioni sottolineate nel caso delle variabili semplici numeriche.

<u>Nota</u>: Per calcolare lo spazio di memoria principale occupato da una variabile multipla numerica si deve utilizzare una delle seguenti formule:

1. Per le variabili multiple i cui elementi sono variabili con indice numeriche in singola precisione:
   N = 10 byte + n * 4 byte.

2. Per le variabili multiple i cui elementi sono variabili con indice numeriche in doppia precisione: N = 10 byte + n * 8 byte.

   Dove N è il numero di byte occupato in memoria principale dalla variabile multipla ed n è il numero di elementi che la compongono.

Variabili multiple stringa

Una variabile multipla stringa contiene come elementi delle variabili con indice stringa e può avere una o due dimensioni. Una variabile multipla stringa è indicata con un nome costituito da una lettera maiuscola dell'alfabeto e dal segno di dollaro: A$ può essere il nome di una variabile multipla stringa mentre A1$ non può esserlo.

In un programma si può assegnare lo stesso nome ad una variabile semplice stringa e ad una variabile multipla stringa: così R$ può indicare nello stesso programma una variabile semplice stringa ed una variabile multipla stringa.

In un programma non si può, tuttavia, assegnare lo stesso nome ad una variabile multipla stringa ad una dimensione e ad una variabile multipla stringa a due dimensioni. Il massimo numero di variabili multiple stringa presenti in un programma è 26.

Le singole componenti di una variabile multipla stringa sono richiamate nelle singole istruzioni di un programma con il nome della variabile multipla seguito da un indice numerico (se la variabile multipla è ad una dimensione) o da due indici numerici (se la variabile multipla è a due dimensioni). Ad esempio, nel seguente programma:

```
10 DIM A$(20), B$(12,12)
-  -  -
-  -  -
-  -  -
100 PRINT A$ (1), B$(1,1)
-  -  -
-  -  -
-  -  -
9999 END
```

Dopo aver dichiarato 20 componenti per la variabile multipla stringa ad una dimensione A$ e 144 componenti per la variabile multipla stringa a due dimensioni B$,

l'istruzione 100 stampa il contenuto della prima componente di A$ ed il contenuto della prima componente della prima riga di B$.

Quando inizia l'esecuzione di un programma, tutte le variabili con indice stringa sono inizializzate con il valore "non definito" e valgono le medesime considerazioni già sottolineate nel caso delle variabili semplici stringa (vedi paragrafo "Variabili semplici stringa").

Nota: Per calcolare lo spazio di memoria principale occupato da una variabile multipla stringa, si deve utilizzare la seguente formula: N = 13 byte + n*(a+2) dove N è il numero di byte occupato in memoria principale della variabile multipla, n è il numero di componenti della variabile multipla ed a è la dimensione di allocazione dichiarata esplicitamente od implicitamente per ogni elemento della variabile multipla.

Nomi delle variabili

Riassumiamo nella tabella 4-2 le regole per la generazione dei nomi delle variabili in un programma BASIC.

Variabile	Nome	Esempi
Variabile semplice numerica	Alfa [Cifra]	Z, JØ, Z9
Variabile semplice stringa	Alfa [Cifra] $	Z$, JØ$, Z9$
Variabile multipla numerica	Alfa	Z, J
Variabile multipla stringa	Alfa $	Z$, J$
Variabile con indice numerica	Alfa (Indice[,Indice])	Z(12),J(14,7)
Variabile con indice stringa	Alfa $ (Indice[,Indice])	Z$(5), J$(8,9)

Dove: Alfa è una lettera maiuscola dell'alfabeto inglese (da A a Z)
Cifra è una cifra decimale (da Ø a 9)
Indice è una espressione numerica che, arrotondata all'intero più prossimo, deve fornire un numero maggiore di zero (nel caso di due indici, il prodotto dei due non deve essere maggiore del prodotto 256*256).

Tabella 4-2 Regole per la generazione dei nomi delle variabili

Area comune

I valori assegnati a variabili semplici e variabili multiple possono essere scambiati fra programmi distinti utilizzando un'istruzione COMMON (vedi capitolo 5).

Funzioni di sistema

Il linguaggio BASIC P6066 fornisce la possibilità di utilizzare nelle istruzioni alcune funzioni di sistema di tipo numerico e stringa. Le funzioni di sistema sono richiamabili nelle singole istruzioni di programma mediante un nome e, se richiesto, uno o più argomenti posti tra parentesi. Gli argomenti sono costanti, variabili o espressioni su cui si applica la funzione stessa. Elenchiamo nei successivi paragrafi i due tipi di funzioni disponibili.

Funzioni numeriche di sistema

Si noti che,oltre alle funzioni di sistema, l'utente può utilizzare delle funzioni numeriche e stringa (funzioni utente) che ha definito nel programma (vedi capitolo 5: istruzioni DEF monolinea e multilinea ed istruzione FKEY#). Una funzione numerica è un algoritmo che, applicato a costanti, variabili od espressioni, che ne costituiscono gli argomenti, fornisce come risultato un numero. Una funzione numerica può essere usata in qualsiasi istruzione in cui possa essere presente una espressione numerica, ad esempio:

```
50 LET Z = SIN(X*PI)
70 PRINT SQR(ABS(Y))
```

Nella tabella 4-3 sono riassunte, in ordine alfabetico, tutte le funzioni numeriche di sistema disponibili; la lettera X indica l'argomento della funzione. Quando l'argomento della funzione COT (X) è π radianti od un suo multiplo, il valore ritornato dalla funzione è + 9.999999999999E+99. Quando l'argomento della funzione TAN (X) è $\frac{\pi}{2}$ + (K*π), con K intero maggiore di 1, il valore ritornato dalla funzione è + 9.999999999999E+99. In entrambi i casi il sistema visualizza un messaggio di errore recuperabile commutando nello stato di debugging.

Si noti che le funzioni DET ed RND non hanno argomento. In effetti nella funzione DET l'argomento è implicito ed è rappresentato dall'ultima matrice quadrata di cui, nel programma, si è calcolata la sua matrice inversa. Per poter eseguire la funzione DET si deve aver prima eseguito un comando OPTIONS specificando l'operando MAT.

Si noti infine che il valore numerico ritornato dalle funzioni numeriche di sistema è sempre rappresentato in doppia precisione.

Nome	Descrizione
ABS(X)	Valore assoluto di X
ACS(X)	Arcocoseno (in radianti) di X
ASN(X)	Arcoseno (in radianti) di X
ATN(X)	Arcotangente (in radianti) di X
COS(X)	Coseno di X radianti
COT(X)	Cotangente di X radianti
DEG(X)	Valore, in gradi, di X radianti
DET	Determinante di una matrice quadrata
EXP(X)	Esponenziale in base e di X
HCS(X)	Coseno iperbolico di X radianti
HSN(X)	Seno iperbolico di X radianti
HTN(X)	Tangente iperbolica di X radianti
INT(X)	Il più grande numero intero minore od uguale ad X
LGT(X)	Logaritmo in base 10 di X
LOG(X)	Logaritmo naturale di X
RAD(X)	Valore, in radianti, di X gradi
RND	Numero casuale compreso tra zero ed uno
SGN(X)	Segno di X (+1 per X positivo, Ø per zero, -1 per X negativo)
SIN(X)	Seno di X radianti
SQR(X)	Radice quadrata di X
TAN(X)	Tangente di X radianti

Tabella 4-3 Funzioni numeriche di sistema

Funzioni stringa di sistema

Una funzione stringa di sistema può essere richiamata in tutte le istruzioni di programma nelle quali può apparire una espressione stringa, ad esempio:

```
50 PRINT CHR$(70)
80 LET A$ = B$ + EXT$(C$,5,8)
```

Nella tabella 4-4 sono riassunte, in ordine alfabetico, le funzioni stringa di sistema disponibili. Le variabili stringa, specificate come argomenti, possono essere, in generale, delle espressioni stringa; le variabili numeriche, specificate come argomenti, possono essere, in generale, delle espressioni numeriche.
Si osservi che le funzioni BLN$ e REP$ possono essere eseguite solamente se prima è stato eseguito un comando OPTIONS specificando l'operando STR.

Nome	Descrizione
BLN$(X,A$,B$)	Permette l'accesso ai bit che costituiscono i caratteri del valore di due espressioni stringa e la loro combinazione mediante un operatore booleano specificato come argomento della funzione. Si veda la completa spiegazione al termine della presente tabella.
CHR$(X)	Converte un numero compreso tra Ø e 255 nel corrispondente carattere del set P6066 (vedi appendice E).
EXT$(A$,I,T)	Fornisce la sottostringa di A$ che inizia dall'I-esimo carattere e termina con T-esimo carattere. Nota: Se I>T, o I≤Ø, o T≤Ø, o T maggiore della lunghezza attuale di A$ il valore di ritorno è la stringa nulla ed il sistema commuta nello stato di debugging segnalando un errore di tipo recuperabile.
LEN(A$)	Fornisce il numero di caratteri che sono contenuti nella variabile A$.
REP$(A$,B$,C$,N,I)	Ritorna il valore della stringa A$ modificato sostituendo la sottostringa B$ con C$, N volte a partire dall'I-esimo carattere.

Note: 1) Se N=Ø non viene effettuata alcuna modifica in A$.

2) Se N<Ø in A$ sono sostituite tutte le occorrenze del valore di B$ con il valore C$ a partire dall'I-esimo carattere.

3) Se N>Ø ed il valore di B$ è la stringa nulla, in A$ viene inserito il valore di C$, N volte a partire dall'I-esimo carattere.

4) Se N<Ø ed il valore di B$ è la stringa nulla, A$ non viene modificata ed il sistema commuta nello stato di debugging.

5) Se il valore di C$ è la stringa nulla ed il valore di N<Ø, allora in A$ sono cancellate tutte le ricorrenze del valore di B$ a partire dall'I-esimo carattere.

6) Se la stringa di caratteri ritornata dalla funzione ha un numero di caratteri superiore alla lunghezza di allocazione di A$, allora il sistema commuta nello stato di debugging troncando la stringa dei caratteri eccedenti la lunghezza suddetta e segnalando un errore recuperabile. Se il primo argomento è una espressione stringa e la stringa di caratteri ritornata dalla funzione ha più caratteri del valore dell'espressione suddetta, il sistema commuta nello stato di debugging troncando la stringa dei caratteri eccedenti e segnalando un errore recuperabile.

7) Se I è maggiore della lunghezza attuale di A$, il valore ritornato dalla funzione è il valore di A$. Nel caso in cui il primo argomento della funzione sia una espressione stringa ed I sia maggiore del numero di caratteri che ne compongono il valore, il valore ritornato dalla funzione è il valore del primo argomento.

SCN(A$,B$,Y,X,)

Se Y>Ø e X>Ø la funzione SCN fornisce la posizione in A$ della Y-esima ricorrenza di B$, a partire dal carattere nella posizione indicata da X. La scansione in A$ è effettuata muovendosi di un carattere alla volta. (Se in A$, a partire dalla X-esima posizione, non è presente il valore di B$ viene ritornato il valore zero).

Se Y<Ø e X>Ø la funzione SCN fornisce la posizione in A$ della Y-esima ricorrenza di B$, a partire dal carattere nella posizione indicata da X. La scansione in A$ è effettuata muovendosi con passi uguali alla

	lunghezza attuale di B$. (Se durante la scansione in A$, a partire dalla X-esima posizione, non è trovato il valore di B$ viene ritornato il valore zero). Se Y>Ø e X<Ø la funzione SCN fornisce la posizione in A$ della Y-esima ricorrenza di una stringa diversa del valore di B$, a partire dal carattere nella posizione indicata dal valore assoluto di X. La scansione in A$ è effettuata muovendosi di un carattere alla volta. (Se in A$, a partire dalla X-esima posizione, non è presente alcuna stringa diversa dal valore di B$, viene ritornato il valore zero). Se Y<Ø e X<Ø la funzione SCN fornisce la posizione della Y-esima ricorrenza di una stringa diversa dal valore di B$, a partire dal carattere nella posizione indicata dal valore assoluto di X. La scansione in A$ è effettuata muovendosi con passi uguali alla lunghezza attuale di B$. (Se, durante la scansione in A$, a partire dalla X-esima posizione, non è trovata alcuna stringa diversa dal valore di B$, viene ritornato il valore zero). Nota: Per qualsiasi funzione SCN, se il valore assoluto di X è maggiore della lunghezza attuale di A$, viene ritornato il valore zero.

Tabella 4-4 Funzioni stringa di sistema

Nel seguito diamo una spiegazione completa della funzione BLN$ ed alcuni esempi d'impiego delle funzioni stringa di sistema definite nella tabella 4-4; le note riferite agli esempi ne ampliano la descrizione.

Funzione BLN$: Il formato generale della funzione è:

BLN$ (num-exp, string-exp$_1$, string-exp$_2$)

dove:

num-exp

è una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica l'operatore booleano con cui sono combinati i bit corrispondenti ai caratteri del valore delle due espressioni stringa

string-exp$_1$ e string-exp$_2$

sono due espressioni stringa.

Il valore ritornato dalla funzione è la stringa di caratteri che si ottiene combinando i bit corrispondenti ai caratteri dei valori delle due espressioni stringa secondo le regole algebriche dell'operatore booleano specificato con il valore, arrotondato all'intero più prossimo, di num-exp. Diamo qui di seguito il significato dei diversi operatori booleani che si possono specificare.

Valore di num-exp	Valore ritornato
0	Stringa con caratteri corrispondenti ad 8 bit uguali a zero
1	"string$_1$" AND "string$_2$"
2	"string$_1$" AND (NOT "string$_2$")
3	"string$_1$"
4	(NOT "string$_1$") AND "string$_2$"
5	"string$_2$"
6	"string$_1$" XOR "string$_2$"
7	"string$_1$" OR "string$_2$"
8	"string$_1$" NOR "string$_2$"
9	NOT ("string$_1$" XOR "string$_2$")
10	NOT "string$_2$"
11	"string$_1$" OR (NOT "string$_2$")
12	NOT "string$_1$"
13	(NOT "string$_1$") OR "string$_2$"
14	("string$_1$" NAND "string$_2$")
15	stringa con caratteri corrispondenti ad 8 bit a 1.

Le virgolette indicano che l'operazione è eseguita considerando i valori delle due stringhe string-exp, e string-exp_2.

Se il valore di num-exp, arrotondato all'intero più prossimo, è minore di zero o maggiore di 15 allora la funzione ritorna una stringa nulla.

Se il valore di num-exp è zero o 15 la stringa ritornata ha la lunghezza della prima.

Se le lunghezze di $string_1$ e $string_2$ sono diverse alla stringa con meno caratteri sono aggiunti caratteri corrispondenti ad 8 bit a zero, in modo da eguagliare la lunghezza delle due stringhe, prima che l'operazione booleana specificata sia eseguita.

Diamo nel seguito le tabelle di verità degli operatori booleani suddetti; all'interno delle caselle sono riportati i valori ottenuti mentre all'esterno sono indicati i valori dei bit su cui si applica il relativo operatore booleano. Nell'appendice E è riportata la corrispondenza tra i caratteri del set ISO P6066 e la relativa rappresentazione su 8 bit.

Operatore AND :

	0	1
0	0	0
1	0	1

Operatore OR :

	0	1
0	0	1
1	1	1

Operatore NOT :

0	1
1	0

Operatore XOR :

	0	1
0	0	1
1	1	0

Operatore NOR :

	0	1
0	1	0
1	0	0

Operatore NAND :

	0	1
0	1	1
1	1	0

Funzione CHR$: Diamo un esempio d'impiego della funzione CHR$ in una routine che permette di stampare tutti i caratteri del set P6066.

```
10 FOR I=0 TO 255
20 PRINT I,CHR$(I)
30 NEXT I
40 END
```

Eseguendola si ha:

```
RUN
**** FORMALLY CORRECT PROGRAM ****
 0                 @
 1                 ?
 2                 ⊥
 3                 ⊔
 4                 ⊂
 5                 ⊇
 6
 7                 ⊃
 8                 ⊆
 9                 ⊢
 10                ⊨
 11                ⊣
 12                ⊤
 13                ⊥
 14                ⊗
 15                ⊘
 16                ⊟
 17                ⊕
 18                ⊖
```

e così via fino al 256-esimo carattere. Per brevità si è riportata solo la parte iniziale della stampa prodotta.

Funzione EXT$: Diamo un esempio d'impiego della funzione EXT$ in una routine che stampa una parte predefinita di una variabile stringa.

```
NEW
10 DCL 21(B$)
20 B$="LUNGHEZZA AREA VOLUME"
30 PRINT TAB(35);EXT$(B$,11,14)
40 END
```

Eseguendo la routine si ha:

```
RUN
**** FORMALLY CORRECT PROGRAM ****
                                  AREA
```

Funzione LEN: Diamo un esempio d'impiego della funzione LEN in una routine che permette di determinare se effettivamente il numero di caratteri di una variabile stringa può essere 1023.

```
NEW
AUT
10 DCL 1023(B$)
20 A$="*Olivetti*"
30 B$=""
40 FOR I=1 TO 102
50 B$=B$+A$
60 NEXT I
70 B$=B$+"END"
80 PRINT "B$ ha";LEN(B$);"caratteri"
90 PRINT B$
100 END
```

Eseguendo la routine si ha:

```
RUN
**** FORMALLY CORRECT PROGRAM ****
B$ ha 1023 caratteri
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*END
```

Funzione REP$: Diamo un esempio di come si può sostituire, utilizzando la funzione REP$, una sottostringa di una stringa.

```
LIST
FILE     REPVIG


0010 DCL 40(A$)
0020 LET A$="AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH"
0025 PRINT A$+" e` A$"
0027 PRINT
0030 PRINT REP$(A$,"HHH","ZZZ",2,7)
0040 PRINT REP$(A$,"HHH","ZZZ",0,7)
0050 PRINT REP$(A$,"HHH","ZZZ",-1,7)
0060 PRINT REP$(A$,"HHH","ZZZ",-100,7)
0070 PRINT REP$(A$,"","ZZZ",3,7)
0080 PRINT REP$(A$,"","ZZZ",-1,7)
0090 PRINT REP$(A$,"HHH","",-1,7)
0100 END

END OF LISTING
```

Eseguendo la routine si ha:

```
RUN
AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH e` A$

AAAHHHBBBZZZCCCZZZDDDHHHEEEHHH
AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH
AAAHHHBBBZZZCCCZZZDDDZZZEEEZZZ
AAAHHHBBBZZZCCCZZZDDDZZZEEEZZZ
AAAHHHZZZZZZZZZBBBHHHCCCHHHDDDHHHEEEHHH
AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH
ERROR 2   IN LINE 80
AAAHHHBBBCCCDDDEEE
```

Come si vede, l'istruzione 30 stampa il contenuto di A$ modificato sostituendo HHH con ZZZ per due volte a partire dal settimo carattere.

Nell'istruzione 40, il numero di volte con cui HHH deve essere sostituito con ZZZ è zero per cui, come si vede, il contenuto di A$ rimane invariato.

Nell'istruzione 50, il numero di volte con cui HHH deve essere sostituito con ZZZ è indicato con un numero negativo, per cui, a partire dal settimo carattere, tutte le ricorrenze di HHH vengono sostituite con ZZZ (vedi la stampa relativa).

Questa è una regola che è valida per qualunque numero negativo, infatti, nella istruzione 60 il numero -1 è stato sostituito con -100 ed il risultato ottenuto è identico al precedente, come si può vedere dalla relativa stampa.

L'istruzione 70 mette in evidenza che avendo indicato come stringa da sostituire la stringa nulla ("""), a partire dal settimo carattere, viene inserita per tre volte la stringa ZZZ in successione.

Nell'istruzione 80, avendo specificato come stringa da sostituire la stringa nulla e come numero di ricorrenza di essa da sostituire un numero negativo, la stringa contenuta in A$ non viene modificata ed il sistema commuta nello stato di debugging. L'operatore può scegliere se continuare nell'esecuzione del programma, premendo il tasto di console CONTINUE, o terminare l'esecuzione e commutare il sistema nello stato comandi, premendo il tasto di console BREAK.

Nella istruzione 90 è stata utilizzata la funzione REP$ per cancellare tutte le ricorrenze della sottostringa HHH nella stringa contenuta in A$ a partire dal settimo carattere. Questo perchè come terzo argomento si è specificata la stringa nulla e come quarto un numero negativo.

Funzione SCN: Vediamo un esempio d'impiego della funzione SCN per rintracciare la presenza di un carattere in una stringa.

```
LIST
FILE     SCN

0010 DCL 80(A$,B$),150(M$)
0020 DISP "INTRODUCI TESTO PER A$ SENZA PUNTO";
0030 INPUT A$
0040 DISP "INTRODUCI TESTO CON PUNTO PER B$";
0050 INPUT B$
0060 LET M$=A$+B$
0070 PRINT "Nel testo vi sono ";SCN(M$,".",1,1);"caratteri."
0080 IF SCN(M$,"!",1,1)=0 THEN 120
0090 DISP "INTRODUCI TESTO CHIUSO CON PUNTO";
0100 INPUT M$
0110 IF SCN(M$," ",1,70)+1=1 THEN 140
0120 PRINT "Il carattere ! non e' nel testo."
0130 GOTO 90
0140 PRINT "Il testo ha meno di 70 caratteri."
0150 END

END OF LISTING
```

Eseguendo la routine si ha:

```
RUN
INTRODUCI TESTO PER A$ SENZA PUNTO?
L'ITALIA E' UNA PENISOLA
INTRODUCI TESTO CON PUNTO PER B$?
NEL SUD DELL'EUROPA.
Nel testo vi sono  44 caratteri.
Il carattere ! non e' nel testo.
INTRODUCI TESTO CHIUSO CON PUNTO?
L'AMERICA E' UN CONTINENTE IMMENSO.
Il testo ha meno di 70 caratteri.
```

Come si vede dall'istruzione 80, se il carattere o la stringa ricercata non esiste nella stringa analizzata, viene fornito come risultato lo zero. Il risultato zero viene dato anche se, vedi istruzione 110, la posizione di inizio della ricerca è superiore al numero di caratteri della stringa.

Vediamo altri esempi della funzione SCN per diversi valori degli argomenti. Se si ha A$ = "AAAAAABAAAAA"

la funzione SCN (A$,"B",1,1) ritorna il valore 7

la funzione SCN (A$,"AA"",2,1) ritorna il valore 2

la funzione SCN (A$,"AA",-2,1) ritorna il valore 3

la funzione SCN (A$,"A",1,-1) ritorna il valore 7

la funzione SCN (A$,"AAAA",1,1) ritorna il valore 4

la funzione SCN (A$,"AAAA",-1,-1) ritorna il valore 5

la funzione SCN (A$,"AB",1,1) ritorna il valore 6

infine la funzione SCN (A$,"AB",-1,1) ritorna il valore Ø.

Funzione speciale di sistema

Il linguaggio BASIC P6066 permette l'impiego, nelle istruzioni DISP e PRINT (vedi capitolo 5), di una funzione speciale di sistema che è richiamata, nelle suddette istruzioni, con il formato:

TAB (num-exp)

La funzione TAB posiziona il pointer del buffer di display od il pointer del buffer di stampa nella posizione il cui valore corrisponde al valore di num-exp arrotondato all'intero più prossimo. Si veda per ulteriori dettagli la relativa descrizione delle istruzioni DISP e PRINT nel capitolo 5.

Espressioni

Una espressione è qualsiasi rappresentazione di un numero o di una stringa. Quindi le costanti, le variabili semplici, le variabili con indice, le funzioni di sistema e le funzioni di utente sono espressioni.

Sono inoltre espressioni le rappresentazioni che si ottengono combinando qualunque degli elementi suddetti (operandi) con particolari simboli detti operatori. Un operatore può specificare:

- una relazione tra gli operandi
- una operazione da eseguire sugli operandi
- se gli operandi sono numeri positivi o negativi

Per esempio i simboli =, * e - sono operatori che specificano, rispettivamente, la relazione uguale, l'operazione di moltiplicazione ed il segno meno. (= e - possono specificare anche, rispettivamente, l'operazione di assegnazione e l'operazione di sottrazione). Una particolare categoria di espressioni, dette espressioni di confronto, è utilizzata nell'istruzione IF per scegliere quale, tra diverse routine, eseguire come conseguenza di un confronto tra dati del programma.

Espressioni numeriche

Una espressione numerica può essere una costante numerica, una variabile semplice numerica, una variabile con indice numerica, un riferimento ad una funzione numerica di sistema od un riferimento ad una funzione numerica definita dall'utente; ma può essere anche una qualunque combinazione degli elementi suddetti (operandi) separati da operatori numerici e parentesi e preceduti dal segno algebrico. Alcuni esempi di espressioni numeriche sono:

```
A
B (5,4)
-25.4
SQR(X)
FNA(Y,Z)
(FNA(Y,Z) * SQR(X) + B (5,4) / (25.4)) - A↑3
```

Il valore di una espressione numerica è ottenuto eseguendo le operazioni specificate, sugli operandi indicati, secondo le regole descritte nel paragrafo successivo.

Note:

1. Se durante l'esecuzione di una espressione numerica si ottiene un risultato intermedio il cui valore è nella zona di OVERFLOW (definita per la rappresentazione in doppia precisione), il sistema continua a valutare l'espressione assumendo come valore intermedio $-9.999999999999*10^{99}$ oppure $9.999999999999*10^{99}$. Il valore finale dell'espressione è assegnato alla variabile specificata nella corrispondente istruzione di assegnazione LET, ma il sistema commuta nello stato di debugging e visualizza un messaggio di errore recuperabile.

2. Se durante l'esecuzione di una espressione numerica si ottiene un risultato intermedio il cui valore è nella zona di UNDERFLOW (definita per la rappresentazione in doppia precisione), il sistema continua a valutare l'espressione assumendo come valore intermedio zero. Il valore finale dell'espressione è assegnato alla variabile della corrispondente istruzione di assegnazione (LET) ma il sistema commuta nello stato di debugging e visualizza un messaggio di errore recuperabile.

3. Se l'indice (o gli indici) di una variabile con in-

dice è dato in forma di espressione numerica, l'espressione viene calcolata e il risultato è arrotondato all'intero più prossimo. Quindi se x è il valore dell'espressione e n la sua parte intera l'indice assumerà il valore:

n	se	$n.0 \leq x \leq n.5$
n+1	se	$n.5 \leq x \leq (n+1).0$

cioè	se	x=5.49	allora n=5
	se	x=5.5	allora n=6

4. L'impiego contemporaneo, in una espressione numerica, di quantità molto piccole e molto grandi (in valore assoluto) può produrre dei risultati inattesi, dovuti al tipo di rappresentazione interna dei dati; così:

```
10 PRINT  2↑63-1-2↑63+1
20 END
RUN
1
```

il risultato ottenuto è dovuto al fatto che nella espressione contenuta nell'istruzione 10 l'esecuzione della sottrazione 2↑63-1 non modifica 2↑63. Cambiando la sequenza si ha:

```
10 PRINT  2↑63-2↑63+1-1
20 END
RUN
0
```

quindi il risultato esatto.

Operatori numerici

Gli operatori numerici definiscono quale operazione deve essere eseguita sui valori numerici degli operandi specificati. Essi producono come risultato un numero. Gli operatori numerici che si possono utilizzare sono:

Operatore numerico	Funzione
↑	Elevamento a potenza
/	Divisione
*	Moltiplicazione
+	Addizione e segno più
-	Sottrazione e segno meno

Le espressioni numeriche sono eseguite secondo il livello di priorità degli operatori che le costituiscono. Le operazioni con il più alto livello di priorità sono eseguite per prime; quelle con lo stesso livello di priorità sono eseguite nell'ordine da sinistra a destra.

Il linguaggio BASIC osserva le regole dell'algebra per la definizione del livello di priorità di esecuzione di una operazione nell'ambito di una espressione numerica, per cui i livelli di priorità sono:

Operatore numerico	Livello di priorità
↑	il più alto
* e /	↓
+ e -	il più basso

Le parentesi (e) possono essere usate per cambiare l'ordine di esecuzione delle operazioni nell'ambito di una espressione numerica. Una espressione racchiusa tra parentesi è trattata come un singolo elemento numerico: viene valutata per ottenere il suo valore numerico, quindi tale valore è utilizzato per la valutazione della parte restante di una espressione più complessa di cui essa fa parte. Se più di una espressione è compresa tra parentesi, il calcolo inizia con la valutazione delle parentesi più interne. Nel seguito diamo ulteriori informazioni relative agli operatori numerici.

Elevamento a potenza:

B↑E	indica che il valore di B deve essere elevato all'esponente E
A↑B↑C	viene eseguita come se fosse (A↑B)↑C
Se B=Ø ed E<Ø	si ha una segnalazione di OVERFLOW
Se B<Ø ed E non è intero	si ha una segnalazione di errore recuperabile
Se B=Ø ed E=Ø	B↑E è uguale ad uno
Se B=Ø ed E>Ø	B↑E è uguale a zero

Per calcolare la radice N-esima di un numero positivo si deve eseguire l'espressione X↑(1/N). Se X è negativo viene segnalato un errore recuperabile ed il si-

stema commuta nello stato di debugging.

Moltiplicazione ed addizione:

A*B	equivale a B*A
A+B	equivale a B+A

come si vede, per la moltiplicazione e l'addizione vale la proprietà commutativa.

A*(B*C)	non equivale sempre a (A*B)*C
A+(B+C)	non equivale sempre a (A+B)+C

perchè l'operazione tra parentesi in alcuni casi può dare un risultato arrotondato o troncato.

Divisione e sottrazione:

A/B	indica A diviso per B
se B=Ø	si ha segnalazione di OVERFLOW
A-B	indica A-B

Segno:

-B+(-A)+C-(-Z)	è ammesso
A+-B	non è ammesso

Il segno + ed il segno - possono essere usati dopo una parentesi aperta prima di una espressione numerica. Si noti che avendo l'operatore ↑ un più alto livello di priorità rispetto al segno le due espressioni seguenti non hanno lo stesso significato: -5↑3.2 (-5)↑3.2

Quando è eseguita la prima espressione il risultato ottenuto è -172.46621, mentre quando è eseguita la seconda espressione il risultato ottenuto è 172.46621 ed il sistema è nello stato di debugging mentre sul display appare un messaggio di errore recuperabile.

Espressioni stringa e operatori

Una espressione stringa è costituita da qualsiasi costante stringa, variabile semplice stringa, variabile con indice stringa, funzione stringa di sistema o definita dall'utente; oppure può essere una qualsiasi sequenza degli elementi suddetti (detti operatori) separati da parentesi e da un operatore binario.

L'unico operatore binario è il simbolo di concatenazione + che ad esempio in D$=A$+B$ produce come risultato, in D$, una stringa che è composta dai caratteri di A$ e quelli di B$ aggiunti in sequenza. Quindi la lunghezza della stringa risultante da concatenazione di una o più stringhe è uguale alla somma della lunghezza delle singole stringhe. Esempi di espressioni stringa sono:

```
A$
"Area di base"
REP$ (A$, B$, C$, 3, 1)
A$+"Area di base"+REP$(A$,B$, C$,3,1)
```

Espressioni di confronto

Una espressione di confronto paragona il valore di due espressioni numeriche o stringa. Il risultato del confronto è un valore di verità (vero o falso). Gli operatori di confronto sono:

Operatore	Significato
=	Uguale
<> oppure ><	Non uguale
>= oppure =>	Maggiore o uguale di
<= oppure =<	Minore o uguale di
>	Maggiore di
<	Minore di

Il formato generale della espressione di confronto è:
e1 operatore-confronto e2

e1 ed e2 possono essere qualunque espressione meno che una espressione di confronto. Solo due espressioni si possono confrontare in una espressione di confronto. Le espressioni da confrontare debbono essere entrambe numeriche od entrambe stringa. Nelle espressioni di confronto tra stringhe il paragone avviene carattere per carattere da sinistra a destra; per giudicare se un carattere è maggiore di un altro, si deve osservare la appendice E in cui sono riportati tutti i caratteri del SET P6066 ed i relativi valori decimali; fra due caratteri è maggiore quello a cui corrisponde, nella suddetta tabella, un numero decimale maggiore. Così si ha che le seguenti espressioni di confronto sono vere:

"A" < "B"
"ABC" < "ABD"
"ABC" < "CAB"

Quando si confrontano due stringhe con lunghezza diversa il confronto avviene tra la stringa più corta e la parte di sinistra della stringa più lunga. Se il confronto dà risultato di eguaglianza viene considerata maggiore la stringa più lunga. Così si ha che le seguenti espressioni di confronto sono vere:

ABC < ABCD
HAZ < HAZL
ZAB > ABCDE

Espressioni con variabili multiple numeriche

Una espressione può contenere delle variabili multiple numeriche (matrici); in questo caso l'espressione suddetta appare alla destra del segno uguale. Alcune istruzioni in cui compaiono delle intere matrici allo interno di espressioni sono:

```
10 MAT A = B
20 MAT B = C + D
30 MAT Z = C - D
40 MAT A = (5)*B
```

Per ulteriori informazioni si vedano le istruzioni MAT nel capitolo 5.

File dati

Con il linguaggio BASIC si possono generare ed elaborare insiemi di dati di tipo numerico e/o stringa che prendono il nome di file dati. I file dati sono distinti in file dati interni e file dati esterni.

Si definiscono file dati interni quegli insiemi di dati numerici e/o stringa che sono interni ad un programma e quindi tutti presenti in memoria principale contemporaneamente al programma che li elabora.

Si definiscono file dati esterni quegli insiemi di dati numerici e/o stringa che sono registrati in una libreria e sono richiamati in memoria principale nelle parti che devono essere elaborate. Essi possono essere quindi utilizzati da diversi programmi e possono contenere più dati dei precedenti,poichè la loro dimensione non dipende dalla capacità della memoria principale.

Nei paragrafi successivi vengono analizzati separatamente i due tipi di file suddetti.

File dati interni

I file dati interni sono costituiti da sequenze di dati numerici e/o stringa definite in un programma BASIC. Per definire un file dati interno si utilizza la istruzione DATA. L'istruzione:

```
10 DATA 15, "AREA", 17, "VOLUME"
```

definisce un file dati interno i cui elementi sono in sequenza: il numero 15, la stringa AREA, il numero 17 e la stringa VOLUME. Si possono utilizzare più istruzioni DATA in un programma; in questo caso il file dati corrispondente è costituito da tutti i valori numerici e stringa espressi nelle istruzioni DATA ed ordinati nella sequenza con sui sono presenti, da sinistra a destra, nelle istruzioni stesse. Quindi le istruzioni:

```
10 DATA  1,2,3,4
20 DATA  5,6,7,8
30 DATA  9,A,B,C
40 DATA  D,E,F,G,H,I,L,M,N,O,P,Q
50 DATA  R,S,T,U,V,W,X,Y,Z
```

definiscono un file dati interno ad un programma che inizia con il dato numerico 1 e prosegue poi in sequenza con le cifre da 2 a 9 e quindi con le lettere dell'alfabeto da A a Z. Quindi A sarà il decimo elemento del suddetto file.

Per accedere ai dati del file così definito si utilizza l'istruzione READ, che permette di assegnare alle variabili di programma i valori contenuti nel file suddetto. Nell'esempio seguente:

```
10 DATA  1,2,3,4
20 DATA  5,A,B,C
30 READ  A,B,C
40 READ  D
50 READ  E,A$,B$,C$
```

dopo aver definito il file dati i cui elementi sono in sequenza 1,2,3,4,5,A,B e C, con le istruzioni 30,40 e 50, si assegnano tali valori nell'ordine alle variabili: A,B,C,D,E,A$,B$ e C$.

L'assegnazione dei dati alle variabili è fatta in modo

dinamico ossia, mentre l'ordine con cui si succedono i dati nel file è determinato dall'ordine con cui i dati compaiono nelle istruzioni DATA (da sinistra a destra) e dall'ordine con cui le istruzioni DATA compaiono (secondo il numero di linea) nel programma, l'ordine di assegnazione coincide con l'ordine con cui sono eseguite le istruzioni READ e nell'ambito dell'istruzione READ l'assegnazione procede dalla variabile più a sinistra verso destra. Il file così prodotto è un file sequenziale; infatti i dati sono assegnati alle variabili uno dopo l'altro.

Si può riprendere l'assegnazione dei dati alle variabili dall'inizio del file utilizzando l'istruzione RESTORE. In un programma si può avere un solo file dati interno.

Per ulteriori informazioni sui file dati interni si vedano nel capitolo 5 le istruzioni: DATA, READ e RESTORE.

File dati esterni

I file dati esterni sono insiemi di dati numerici e/o stringa che sono registrati in una libreria.
Questo permette l'impiego di archivi di dati che non sono limitati dalla capacità della memoria principale.
Riferendosi al metodo di elaborazione i file dati esterni si distinguono in:

- file ad accesso sequenziale
- file ad accesso diretto

Un file è ad accesso sequenziale se per accedere ad un suo dato si devono prima leggere tutti i dati che lo precedono sul supporto su cui il file è registrato.

Un file è ad accesso diretto se si può accedere direttamente ad un suo qualsiasi dato senza dover leggere i dati che lo precedono.

Un file dati esterno non può essere generato da tastiera, ma deve essere creato eseguendo un comando CREATE (vedi capitolo 3) ed un programma BASIC. Il comando CREATE alloca lo spazio per il file; il programma genera i dati che saranno registrati nel file. Per ottenere informazioni relative ad un file dati esterno, quale il modo in cui può essere elaborato, lo spazio che occupa etc. si deve eseguire il comando CATALOG.

Creazione di un file dati esterno: Per generare un file dati esterno si deve prima allocare per esso un certo spazio in una libreria utilizzando il comando di sistema CREATE (vedi capitolo 3). Lo spazio da allocare deve essere uguale al numero di byte che si prevede di occupare con i dati del file arrotondato ad un multiplo intero di 128, se il file è creato in una libreria residente su floppy disk, oppure 256 se il file è creato in una libreria residente su disco. Il sistema si riserva in più 128 byte o 256 byte (a seconda se floppy disk o disco) per registrare delle informazioni di servizio. Oltre al numero di byte da riservare al file dati, con il comando CREATE si comunica al sistema il nome del file dati, il tipo di sottolibreria (è implicito nel nome) e la libreria in cui deve essere allocato. I dati sono registrati in un file esterno con un formato diverso da quello con cui sono rappresentati in memoria principale per essere elaborati. Nell'appendice F si possono confrontare le diverse richieste di spazio nei due casi. Per calcolare lo spazio richiesto dal contenuto di un file dati esterno sul suo supporto, si devono tener presenti le seguenti considerazioni:

1. I dati numerici in singola precisione occupano 4 byte.

2. I dati numerici in doppia precisione occupano 8 byte.

3. Le costanti numeriche sono sempre registrate, nel file esterno, in doppia precisione, qualunque sia il loro valore.

4. I dati di tipo stringa occupano 4*INT ((n-1)/4+2) byte, dove INT è la funzione che fornisce la parte intera del valore calcolato tra parentesi ed n è il numero di caratteri che compone la stringa.

Il nome da assegnare ad un file dati può essere costituito da al massimo 7 caratteri, se il file dati deve essere registrato in una sottolibreria package od in una sottolibreria comune; da 6 caratteri, se il file dati deve essere registrato in una sottolibreria utente.

Il primo carattere deve essere asterisco (*) nel caso di un file dati da registrare in una sottolibreria

package, mentre deve essere più (+) nel caso di un file dati da registrare in una sottolibreria comune; in entrambi i casi il secondo carattere deve essere una lettera maiuscola dell'alfabeto inglese (da A a I) ed i restanti caratteri possono essere costituiti da una lettera maiuscola, come il precedente, oppure da una cifra decimale (da Ø a 9).

Nel caso di file dati registrati in una sottolibreria utente il primo carattere deve essere una lettera maiuscola dell'alfabeto inglese ed i restanti caratteri possono essere una lettera, come il precedente, oppure una cifra decimale.

Ricerca di un dato in un file dati esterno: Al fine della ricerca di un dato in un file dati esterno si deve tener presente che i file dati esterni sono suddivisi in parole contigue e ad essi è associato un pointer che indica su quale parola del file può avvenire l'accesso; una parola è costituita da 4 byte; sotto questo aspetto i diversi tipi di dati occupano il seguente spazio:

1. I dati numerici in singola precisione sono contenuti in una parola.

2. I dati numerici in doppia precisione sono contenuti in due parole.

3. Le costanti numeriche sono contenute sempre in due parole.

4. I dati di tipo stringa sono contenuti in INT((n-1)/4+2) parole; dove INT è la funzione che fornisce la parte interna del valore calcolato tra parentesi ed n è il numero di caratteri che compone la stringa. L'ultima parola può non essere completamente occupata dal dato; in questo caso è riempita con spazi aggiunti in coda.

Eseguendo l'istruzione WHERE: (vedi capitolo 5) si può sapere su quale parola del file dati esterno è posizionato il pointer.

Apertura e chiusura di un file dati esterno: Un file si dice aperto nei confronti di un programma se il programma può accedere ad esso. Un file si dice chiuso nei confronti di un programma, se il programma non può

accedere ad esso. Perchè un file possa essere utilizzato da un programma deve essere aperto specificandone il nome in una istruzione FILES. Il numero di file dati che possono essere contemporaneamente aperti all'accesso di un programma,in un certo istante, è limitato dal numero di caratteri che costituiscono l'istruzione FILES.

I file dati aperti all'azione di un progamma mediante l'istruzione FILES possono essere chiusi utilizzando l'istruzione FILE:che può contemporaneamente aprirne degli altri.

Per poter leggere i dati contenuti in un file esterno o registrare in esso dei dati si deve utilizzare l'istruzione FILES (vedi capitolo 5) che assegna ad ogni file specificato in essa 172 byte di memoria principale, se il file è in una libreria residente su floppy disk, oppure 292 byte se il file è in una libreria residente su disco. L'ordine con cui i nomi dei file dati esterni si susseguono nell'istruzione FILES è importante perchè ad esso corrisponde un numero designatore per ogni file che, riportato nelle istruzioni di elaborazione dei file, specifica su quale file deve essere eseguita l'operazione indicata dalla relativa parola chiave.

Con l'istruzione FILE: si possono riassegnare i numeri designatori di file dati esterni a file che non sono stati specificati nell'istruzione FILES (i dettagli sono ampliamente descritti nel capitolo 5 alle istruzioni FILES e FILE).

Elaborazione di un file dati esterno: I file dati esterni di tipo sequenziale, dopo le operazioni preliminari suddette, possono essere letti mediante l'istruzione READ:, i dati letti sono assegnati alle variabili specificate nell'istruzione iniziando dal primo dato ed il pointer indica l'inizio della parola successiva all'ultimo dato letto; in questo modo i dati sono letti in sequenza uno dopo l'altro.

Se si vogliono registrare dei dati dall'inizio del file esterno, si utilizza l'istruzione SCRATCH: e quindi con l'istruzione WRITE: si registrano i dati contenuti nelle variabili in essa specificate.

Se invece si vogliono accodare dei nuovi dati a quelli

esistenti si utilizza l'istruzione APPEND:, il pointer indicherà così l'inizio della parola successiva all'ultimo dato del file esterno e con successive istruzioni WRITE: si potranno registrare nuovi dati in sequenza.

Se dopo l'esecuzione di istruzioni WRITE: si vogliono leggere i dati contenuti nel file esterno si deve prima eseguire l'istruzione RESTORE: che ripone il pointer all'inizio del file e quindi le relative istruzioni READ: leggeranno in sequenza i dati in esso contenuti.

I file dati esterni di tipo ad accesso diretto offrono una maggiore elasticità nelle elaborazioni. Infatti dopo le operazioni preliminari, che sono comunque obbligatorie, eseguite dall'istruzione FILES o da una istruzione FILE:, si possono immediatamente leggere dei dati contenuti nel file o registrare in esso dei dati. Inoltre è possibile leggere un dato posizionato dall'inizio di una parola qualsiasi del file esterno purchè si specifichi prima la posizione di tale parola mediante una istruzione SETW: dopo di che si possono leggere con istruzioni READ: i dati che seguono in sequenza sul file esterno il punto specificato. Naturalmente si possono leggere, volendo tutti i dati in sequenza dall'inizio, eseguendo l'istruzione SETW: specificando la prima parola del file, e quindi le relave istruzioni READ:.

Anche la registrazione può avvenire dall'inizio di una parola qualunque del file dati esterno; questa viene specificata, come suddetto, con l'istruzione SETW:, dopo di che si registrano i dati con le istruzioni WRITE:.

Per aggiungere dati alla fine di quelli già presenti in un file dati esterno non si può utilizzare, nel caso di file ad accesso diretto, l'istruzione APPEND:, ma si può specificare tale posizione con l'istruzione SETW:. Naturalmente si possono registrare, volendo, i dati in sequenza dall'inizio del file esterno eseguendo l'istruzione SETW: specificando la prima parola del file (oppure l'istruzione RESTORE:), e quindi le relative istruzioni WRITE:.

Nota: Nel caso di file esterni, i dati da registrare sono memorizzati in un buffer in memoria principale prima di essere trasferiti nel file esterno. I dati sono trasferiti solamente quando (1) il buffer è pieno, (2) è eseguita una istruzione FILE: oppure END, (3) il tasto BREAK è premuto. Quindi se viene a mancare la tensione di alimentazione alla macchina durante l'esecuzione di un programma che registra dati su un file esterno, alcuni dati possono essere perduti ed il file può restare aperto. Si può determinare se un file dati è rimasto aperto eseguendo il comando CATALOG. (Per chiudere tale file si esegua il comando VALIDATE).

5. LE ISTRUZIONI BASIC

Nel presente capitolo diamo una spiegazione dettagliata di tutte le istruzioni del linguaggio BASIC corredata di esempi di impiego. Il capitolo è stato scritto con l'intenzione di fornire la possibilità di un rapido riferimento ai punti riguardanti l'esatta codifica delle istruzioni e l'impiego corretto delle medesime nell'ambito di un programma BASIC. Nei paragrafi precedenti tale descrizione sono richiamate le nozioni fondamentali che permettono la lettura delle pagine successive che sono, insieme al capitolo 3, le parti più importanti del manuale.

Il programma BASIC e le istruzioni BASIC

Un programma BASIC è composto da un insieme di istruzioni l'ultima delle quali è sempre una istruzione END. Le istruzioni BASIC si classificano in:

- istruzioni eseguibili
- istruzioni non eseguibili

Le istruzioni eseguibili specificano al sistema di compiere una azione ben determinata; sono esempi di istruzioni eseguibili: l'istruzione LET che assegna un valore ad una o più variabili, l'istruzione DISP che visualizza il contenuto delle variabili di programma in essa specificate, l'istruzione GOSUB che modifica l'ordine sequenziale di esecuzione di un programma.

Le istruzioni non eseguibili si limitano a specificare delle informazioni che sono utili per l'esecuzione del programma o per il programmatore; sono esempi di istruzioni non eseguibili: l'istruzione DIM che specifica le dimensioni di una variabile multipla, l'istruzione DCL che specifica l'occupazione di memoria principale di una variabile, l'istruzione REM che specifica un commento utile per il programmatore che è stampato quando si richiede il listing del programma.

Le istruzioni non eseguibili possono essere interposte con le istruzioni eseguibili nell'ambito del programma,

ma per facilitarne la leggibilità è bene raggruppare tutte le istruzioni di tipo dichiarativo (DIM, DCL) in un'unica parte del programma.

Ogni istruzione in un programma BASIC è detta linea BASIC e deve iniziare con un numero di linea che è costituito da un numero intero compreso tra 1 e 9999. Il numero di linea determina l'ordine di esecuzione delle istruzioni del programma; ad esso possono riferirsi altre istruzioni di programma (vedi GOTO, GOSUB etc.) oppure dei comandi di sistema (vedi LIST, FETCH etc.). Tutte le istruzioni sono eseguite in ordine di numero di linea, prescindendo dall'ordine con cui sono state introdotte, a meno che la sequenza di esecuzione sia alterata da salti od iterazioni (vedi le istruzioni GOTO, GOSUB, FOR/NEXT etc.).

Introduzione di linee BASIC

Le linee BASIC sono introdotte da tastiera ed ognuna può avere al massimo 80 caratteri; ogni linea è trasferita in memoria principale premendo il tasto [END OF LINE] (vedi capitolo 2). Non vi possono essere due linee con lo stesso numero di linea in uno stesso programma. In questo caso l'ultima linea digitata è inserita nel programma, mentre la precedente è cancellata.

Una linea BASIC in generale è composta, oltre che dal numero di linea, da:

- una o più parole chiave BASIC
- uno o più operandi

Le parole chiave BASIC sono parole inglesi con lettere maiuscole (vedi FOR, STEP) o caratteri speciali (vedi due punti ":" per l'istruzione immagine) che specificano al sistema l'azione o le azioni da eseguire. Gli operandi possono specificare:

- su quali elementi di programma (costanti, variabili o espressioni) deve essere compiuta l'azione espressa dalle parole chiave BASIC, ad esempio:
  100 PRINT"OLIVETTI", A,A*B, A$

- quali condizioni (relazione di confronto) si devono verificare perchè l'azione espressa dalla parola chiave BASIC si attui, ad esempio:
  50 IF A=B THEN 1000
  dove A=B indica la condizione che si deve verificare

perchè l'esecuzione del programma prosegua dalla istruzione con numero di linea 1000

- con quali modalità l'azione espressa dalla parola chiave BASIC si deve attuare, ad esempio:
  70 DELAY 100
  dove 100 indica che l'esecuzione dell'istruzione successiva deve essere ritardata di 10 secondi

Una linea BASIC può essere digitata senza introdurre spazi tra le parti (numero di linea, parola chiave BASIC ed operandi) che la compongono. Quando però si esegue una stampa (vedi comando LIST) od una visualizzazione (vedi comando FETCH) delle istruzioni di un programma, le istruzioni sono stampate e visualizzate con un formato diverso da quello con cui sono state introdotte per poter permettere una facile lettura del programma stesso.

Le parole chiave BASIC possono essere digitate da tastiera premendo un solo tasto della sezione alfanumerica mentre si mantiene premuto il tasto [SHIFT]. Questo rende più rapida l'introduzione da tastiera delle istruzioni BASIC e riduce le possibilità di errore. Dopo l'introduzione l'istruzione BASIC è analizzata. Se è rilevato un errore sintattico viene immediatamente visualizzato un messaggio di errore; altrimenti la istruzione è trasferita in memoria principale nell'ambito del programma.

Notazioni

Le seguenti notazioni sono impiegate nella descrizione delle istruzioni BASIC:

{ } racchiude due o più parametri che non sono opzionali; uno di essi deve essere specificato

[ ] racchiude uno o più parametri che sono opzionali, un parametro o nessun parametro può essere specificato

... indica che il precedente operando può essere ripetuto più di una volta

, separa gli operandi di una istruzione BASIC (nelle istruzioni PRINT, MAT PRINT e DISP ha inoltre una funzione di tabulazione standard)

I seguenti simboli sono usati per definire il formato delle istruzioni, ma non devono essere digitati:

- trattino di unione
{ } parentesi graffe
[ ] parentesi quadre
... puntini

Le lettere minuscole e le parole con lettere minuscole rappresentano informazioni variabili che deve fornire l'utente. Le parole con lettere maiuscole (parole chiave BASIC) ed i seguenti simboli devono essere digitati esattamente come indicato nella definizione dell'istruzione:

segno di numero
* asterisco, prodotto scalare o prodotto tra matrici
$ segno di dollaro
: due punti
; punto e virgola
" apici
( ) parentesi tonde
+ addizione
- sottrazione

Elenco e funzione delle istruzioni BASIC

Le istruzioni BASIC e la loro funzione sono elencate in ordine alfabetico, con tutte le istruzioni per il calcolo sulle matrici nell'ultima parte, come segue:

Istruzione	Funzione
APPEND:	Permette di aggiungere dati in coda ad un file dati esterno, sequenziale
ASSIGN	Assegna ad una o più variabili uno o più dati contenuti in una stringa di caratteri e separati da un delimitatore
BASSIGN	Assegna ad una o più variabili di programma le stringhe e/o i dati numerici compresi nel valore di una espressione stringa
BBUILD	Trasferisce il valore di una o più espressioni ad una variabile stringa, modificandone il formato
BEEP	Produce una segnalazione acustica
BPAD	Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione, aggiungendo in coda dei caratteri binari

Istruzione	Funzione
BUILD	Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa
BUILD USING	Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa ponendo i caratteri in posizioni predefinite da una istruzione immagine
CALL	Carica in memoria principale il modulo assembler specificato e ne inizia l'esecuzione
CHAIN	Termina l'esecuzione di un programma presente in memoria principale e carica in memoria principale il programma specificato lanciandone l'esecuzione
CONVERT	Converte ogni carattere di una espressione stringa nel corrispondente codice numerico ISO e viceversa
DATA	Crea un file dati interno al programma
DCL	Dichiara la lunghezza di allocazione delle variabili stringa e la singola precisione per le variabili numeriche
DEF	Definisce una funzione monolinea
DEF/FNEND	Definiscono una funzione multilinea
DELAY	Ritarda l'esecuzione dell'istruzione successiva
DEPAD	Rimuove in una variabile stringa i caratteri di riempimento specificati
DIM	Specifica le dimensioni delle variabili multiple di programma
DISP	Visualizza dati e testi sul display in formato standard
DISP USING	Visualizza dati e testi sul display in un formato predefinito in una istruzione immagine
END	Definisce la fine di un programma
FILES	Specifica quali file dati esterni possono essere elaborati dal programma

Istruzione	Funzione
FILE:	Chiude ed apre l'accesso di un programma ad un file dati esterno
FKEY #	Assegna un contenuto ai tasti funzione
FNEND	Termina la definizione di una funzione multilinea
FOR	Inizia l'esecuzione di un ciclo iterativo
GOSUB	Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma
GOTO	Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata
IF...THEN	Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata, nel caso che si verifichi la condizione predefinita
Istruzione Immagine	Specifica un formato predefinito utilizzato dalle istruzioni PRINT USING, DISP USING, BUILD USING, MAT PRINT USING
INPUT	Assegna i valori introdotti da tastiera alle variabili di programma specificate
INTERRUPT ENABLE	Consente la gestione da parte dell'utente di interruzioni sia interne che da periferiche esterne
LET	Assegna valori alle variabili di programma
NEXT	Definisce il termine di un ciclo iterativo
ON...GOSUB	Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma scelto tra un insieme di sottoprogrammi in funzione del valore assunto da una espressione specificata
ON...GOTO	Trasferisce il controllo dell'esecuzione di un programma ad una istruzione scelta tra un insieme di istruzioni in funzione del valore assunto da una espressione specificata
PAD	Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione aggiungendo in coda dei caratteri predefiniti
PRINT	Stampa dati e testi in un formato standard

Istruzione	Funzione
PRINT USING	Stampa dati e testi in un formato predefinito in una istruzione immagine
RANDOMIZE	Permette la generazione di numeri casuali
READ	Assegna alle variabili di programma specificate i valori contenuti nel file dati interno al programma, generato con le istruzioni DATA
READ:	Assegna alle variabili di programma specificate i valori contenuti in un file dati esterno
REMARK	Permette di inserire in un programma dei commenti che rendono facile la lettura del relativo listing
RESTORE	Posiziona il pointer del file dati interno all'inizio del file stesso
RESTORE:	Posiziona il pointer di un file dati esterno all'inizio del file e, se il file è di tipo sequenziale, ne permette la lettura
RETURN	Trasferisce il controllo della esecuzione di un programma all'istruzione successiva ad una istruzione GOSUB
RKB	Assegna ad una variabile stringa i caratteri introdotti da tastiera che sono scelti tra i caratteri del SET P6066
SCRATCH:	Posiziona il pointer di un file dati esterno all'inizio di un file sequenziale e permette di registrare in esso dei dati
SETW:	Posiziona il pointer all'inizio della parola specificata di un file dati esterno ad accesso diretto
STOP	Interrompe l'esecuzione di un programma e commuta il sistema nello stato di debugging
TRACE OFF	Termina la stampa dei numeri di linea delle istruzioni di programma eseguite
TRACE ON	Richiede la stampa dei numeri di linea di ogni successiva istruzione di programma eseguita

Istruzione	Funzione
WHERE:	Determina la posizione su cui è posizionato il pointer di un file dati esterno nell'ambito del file
WRITE:	Registra in un file dati esterno i valori delle espressioni specificate
MAT...=	Assegna i valori degli elementi di una matrice agli elementi di un'altra matrice
MAT...+	Esegue l'operazione di addizione tra due matrici e ne assegna il risultato alla matrice specificata prima del segno uguale
MAT...-	Esegue l'operazione di sottrazione tra due matrici e ne assegna il risultato ad una matrice specificata
MAT...* (scalare)	Moltiplica ogni elemento di una matrice per il valore di una espressione numerica e ne assegna il risultato ad un'altra matrice specificata
MAT...*	Esegue il prodotto, righe per colonne, tra due matrici e ne assegna il risultato ad una matrice specificata
MAT...CON	Assegna il valore uno ad ogni elemento di una matrice
MAT...IDN	Assegna il valore uno a tutti gli elementi della diagonale principale di una matrice quadrata ed il valore zero a tutti gli altri elementi della matrice
MAT...INV	Calcola la matrice inversa di una matrice quadrata e la assegna ad una matrice specificata
MAT...TRN	Assegna ad una matrice specificata gli elementi di un'altra matrice scambiandone le righe con le colonne
MAT...ZER	Assegna il valore zero a tutti gli elementi di una matrice
MAT INPUT	Assegna agli elementi di una variabile multipla i dati introdotti da tastiera
MAT PRINT	Stampa i valori degli elementi di una o più variabili multiple nel formato standard
MAT PRINT USING	Stampa i valori degli elementi di una o più variabili multiple in un formato predefinito in una istruzione immagine

Istruzione	Funzione
MAT READ	Assegna agli elementi di una o più variabili multiple i dati contenuti nel file interno definito mediante le istruzioni DATA
MAT READ:	Assegna agli elementi di una o più variabili multiple i dati contenuti in un file dati esterno
MAT WRITE:	Registra in un file dati esterno i valori degli elementi di una o più variabili multiple specificate.

Descrizione delle istruzioni BASIC

Nel seguito sono descritte dettagliatamente tutte le istruzioni del linguaggio BASIC. Tutte le istruzioni sono disposte in ordine alfabetico, meno le istruzioni che permettono di elaborare le matrici che sono raggruppate nell'ultima parte del paragrafo. Alcune istruzioni sono descritte insieme, poichè sono sempre utilizzate in combinazione, comunque vengono riportati gli eventuali riferimenti. La descrizione di ogni istruzione è realizzata secondo la medesima struttura che è composta dai seguenti punti fondamentali:

- funzione : è una breve descrizione della funzione dell'istruzione
- formato : è una descrizione sintetica del formato più generale dell'istruzione
- azione : è una descrizione dettagliata delle azioni eseguite dal sistema quando l'istruzione è eseguita
- note : è un insieme di osservazioni e di regole da osservare per un corretto impiego dell'istruzione
- esempi : è una raccolta di esempi d'impiego della istruzione descritta che permette di chiarire eventuali dubbi riguardo la codifica corretta dell'istruzione. Per motivi di spazio gli esempi sono brevi e non coprono applicazioni significative.

Istruzione APPEND:

Funzione

Permette di aggiungere dati in coda ad un file dati esterno di tipo sequenziale

Formato

APPEND: file-designator

dove:
file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di file.

Azione

Il puntatore del file specificato con file-designator è posizionato all'inizio della parola successiva all'ultimo dato contenuto nel file dati esterno.

Il file suddetto è aperto al programma per operazioni di registrazione.

Note

1. Il valore di file-designator, arrotondato all'intero più prossimo, deve essere maggiore di zero e minore od uguale al numero di file specificati nella istruzione FILES.

2. Il designatore di file è un numero che indica su quale file dati esterno deve essere fatta l'operazione specificata (vedi istruzioni FILES e FILE:).

3. Il designatore di file dell'istruzione APPEND: deve indicare un file dati di tipo sequenziale.

Esempi

1. Si crei un file sequenziale di 256 byte nella libreria MAT sul floppy disk nell'unità FDU1. Si esegua il programma indicato nel listing sottostante. Il programma registra sul file APPEND i numeri

interi da 1 a 10 e quindi li legge e stampa con la tabulazione standard. Con l'istruzione 130 il pointer si pone in coda al file per cui la stringa successiva viene registrata, istruzione 140, subito dopo il numero 10, come si vede dalla relativa stampa del contenuto del file che è fatta da programma.

```
CRE APPEND,(MAT,FDU1),,256

LIST
FILE     ZA


0010 DCL 80(I$)
0020 FILES APPEND
0030 SCRATCH :1
0040 FOR I=1 TO 10 STEP 1
0050 WRITE :1,I
0060 NEXT I
0070 RESTORE :1
0080 FOR I=1 TO 10 STEP 1
0090 READ :1,I
0100 PRINT I,
0110 NEXT I
0120 PRINT
0130 APPEND :1
0140 WRITE :1,"COME SI VEDE QUESTA STRINGA E' IN CODA AL FILE"
0150 RESTORE :1
0160 FOR I=1 TO 10 STEP 1
0170 READ :1,I
0180 PRINT I,
0190 NEXT I
0200 READ :1,I$
0210 PRINT I$;
0220 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
 1              2              3              4              5
 6              7              8              9              10

 1              2              3              4              5
 6              7              8              9              10
COME SI VEDE QUESTA STRINGA E' IN CODA AL FILE
```

Istruzione ASSIGN

Funzione

Assegna ad una o più variabili uno o più dati contenuti in una stringa di caratteri e separati da un delimitatore.

Formato

ASSIGN string-exp, { **num-var** / **string-var** } [, { **num-var** / **string-var** }] ... ; **delimiter**

dove:

string-exp
è una espressione stringa il cui valore può essere composto da uno o più dati separati da un delimitatore

num-var
è una variabile numerica alla quale viene assegnato un valore numerico contenuto in string-exp

string-var
è una variabile stringa alla quale è assegnato un valore contenuto in string-exp

delimiter
è un numero intero compreso tra zero e 255 che indica quale carattere della tabella ISO (vedi appendice E) separa i diversi dati contenuti nel valore di string-exp.

Azione

L'espressione stringa viene calcolata ed il valore ottenuto viene considerato composto da diversi dati ognuno dei quali è separato dal successivo dal carattere che corrisponde, nella tabella ISO, al numero indicato con delimiter.

Ogni dato viene trasformato nel formato elaborabile dal sistema e quindi assegnato nell'ordine, da sinistra a destra, alle variabili specificate dopo string-exp.

Note

1. Un dato stringa non può essere assegnato ad una variabile numerica.

2. Il numero di dati che compongono il valore di string-exp deve essere maggiore od uguale al numero di variabili specificate dopo string-exp.

3. Se il valore della espressione stringa string-exp contiene due caratteri consecutivi equivalenti al delimitatore delimiter, il dato assegnato alla variabile corrispondente è la stringa nulla.

4. L'istruzione ASSIGN, insieme all'istruzione BUILD, è utile quando si vogliano trasferire dati tra la memoria principale del P6066 ed una unità periferica esterna (vedi manuale P6066 I/O con periferiche esterne).

Esempi

1. Nel listing sottostante si può vedere un programma che assegna a diverse variabili di programma i dati contenuti in una stringa separati dal carattere spazio (a cui corrisponde il numero 32, nella tabella ISO).

```
LIST
FILE

0010 ASSIGN "LUNGHEZZA 155 METRI ALTEZZA 120 CENTIMETRI",L$,L,M$,A$,A,C$;32
0020 PRINT "L$=";L$,"L=";L,"M$=";M$
0030 PRINT "A$=";A$,"A=";A,"C$=";C$
0040 END

END OF LISTING


RUN
L$=LUNGHEZZA     L= 155          M$=METRI
A$=ALTEZZA       A= 120          C$=CENTIMETRI
```

2. Nel programma precedente si è diminuito il numero di variabili a cui assegnare gli elementi della stringa. Come si vede dalla stampa prodotta alle variabili A e C$ è assegnato rispettivamente il valore zero e stringa nulla; viene segnalato un errore di tipo recuperabile ed il sistema è nello stato di debugging in attesa di una decisione da parte dell'utente.

```
LIST
FILE


0010 ASSIGN "LUNGHEZZA 155 METRI ALTEZZA 120 CENTIMETRI",L$,L,M$,A$;32
0020 PRINT "L$=";L$,"L=";L,"M$=";M$
0030 PRINT "A$=";A$,"A=";A,"C$=";C$
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
L$=LUNGHEZZA     L= 155          M$=METRI
A$=ALTEZZA       A= 0            C$=
ERROR 1   IN LINE 30
```

Istruzione BASSIGN

Funzione

Assegna a una o più variabili di programma le stringhe e/o i dati numerici contenuti nel valore di una espressione stringa e rappresentati nel formato con cui sono registrati in un file dati esterno.

Formato

BASSIGN string-exp, {**num-var** | **string-var**} [, {**num-var** | **string-var**}] ...

dove

string-exp
è una espressione stringa composta di una o più variabili stringa i cui valori sono rappresentati nel formato con cui sono rappresentati in un file dati esterno

num-var
è una variabile numerica alla quale viene assegnato il corrispondente dato numerico contenuto nel valore di string-exp

string-var
è una variabile stringa alla quale viene assegnato il corrispondente dato stringa contenuto nel valore di string-exp

Azione

L'espressione string-exp viene valutata e riconosciuta come composta da uno o più dati ognuno dei quali è rappresentato nel formato con cui i dati sono registrati in un file dati esterno. Ogni dato componente del valore di string-exp viene modificato nel formato elaborabile dal sistema ed assegnato, nell'ordine, da sinistra a destra, alle variabili specificate dopo string-exp.

Note

1. Un dato di tipo stringa non può essere assegnato ad una variabile numerica; un dato di tipo numerico non può essere assegnato ad una variabile stringa.

2. Il numero di variabili specificate deve essere minore od uguale al numero di dati componenti il valore di string-exp.

3. In string-exp non possono comparire delle stringhe di caratteri ISO, ad esempio: "AREA", bensì le variabili che compaiono nella espressione devono avere un contenuto assegnato ad esse da una istruzione BBUILD (vedi istruzione BBUILD).

4. L'istruzione BASSIGN, insieme all'istruzione BBUILD, è utile quando si vogliano trasferire dati tra la memoria principale del P6066 ed una unità periferica esterna senza modificare il formato dei dati in stringhe ISO.

Esempio

1. Nel listing seguente è indicato un programma che mostra come il contenuto delle variabili A$ e B$ viene separato ed assegnato, solo nella parte valore, alle variabili specificate nella istruzione 40.

```
LIST
FILE

0010 DCL 30(A$,B$)
0020 BBUILD A$,"AREA","123",249
0030 BBUILD B$,"VOLUME","10",100
0040 BASSIGN A$+B$,S$,N$,N,V$,M$,M
0050 PRINT "A$=";A$,"B$=";B$
0060 PRINT "S$=";S$,"N$=";N$,"N=";N,"V$=";V$
0070 PRINT "M$=";M$,"M=";M
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
A$=▥⊰▩▩AREA▥⊥▩▩123 ▥I▩▩▩▩▩▥      B$=▥⊥▩▩VOLUME  ▥⊥▩▩10  ▥▩▩▩▩▩▩▥
S$=AREA          N$=123           N= 249           V$=VOLUME
M$=10            M= 100
```

Istruzione BBUILD

Funzione

Trasferisce il valore di una o più espressioni ad una variabile stringa, modificandone il formato.

Formato

BBUILD string-var, { **num-exp** | **string-exp** } [, { **num-exp** | **string-exp** }] ...

dove:

string-var
è una variabile stringa in cui vengono trasferiti i valori delle espressioni specificate

num-exp
è una espressione numerica il cui valore è trasferito nella variabile stringa specificata

string-exp
è una espressione stringa il cui valore è trasferito nella variabile stringa specificata

Azione

Le espressioni specificate sono valutate ed i relativi valori sono assegnati nell'ordine, da sinistra a destra, alla variabile stringa specificata dopo la parola chiave.

Note

1. Ogni valore è assegnato a string-var nel formato con cui è registrato in un file dati esterno per cui l'occupazione sara:

   - 4 byte per i valori numerici in singola precisione

   - 8 byte per i valori numerici in doppia precisione

   - INT((n-1)/4+2)*4 byte per i valori di tipo stringa (dove n è il numero di caratteri che compongono la stringa).

2. L'istruzione BBUILD è utile per trasferire dati dalla memoria principale del P6066 ad una periferica esterna. In questo caso, infatti, i dati numerici devono essere assegnati ad una variabile stringa (con una istruzione del tipo: 50 BBUILD A$, A,B,C che trasferisce i valori di A,B e C in A$) prima di essere trasferiti alla periferica esterna. I dati numerici registrati su un supporto di una periferica esterna sono trasferiti in memoria principale ed assegnati ad una variabile stringa. Dalla variabile stringa possono poi essere assegnati ad una variabile numerica eseguendo una istruzione BASSIGN del tipo: 100 BASSIGN A$,A,B,C con cui tre dati registrati su un supporto e ricevuti in A$ sono assegnati nell'ordine alle variabili A,B e C.

   Si noti che per eseguire il trasferimento di dati tra P6066 e periferica esterna si possono utilizzare anche le istruzioni BUILD ed ASSIGN. Le istruzioni BBUILD e BASSIGN sono eseguite più rapidamente dal sistema perchè l'esecuzione della istruzione BBUILD non modifica il formato dei dati in stringhe ISO.

Esempi

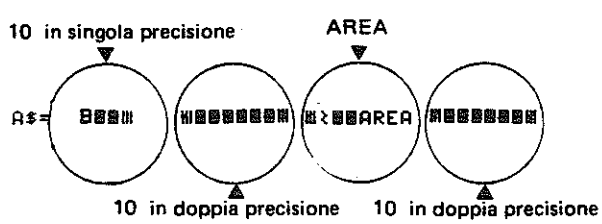
1. Nella stampa prodotta dal programma sottostante si può vedere in quale formato sono rappresentati i dati trasferiti in A$ dall'istruzione BBUILD.

```
FILE      QW

0010 DCL SA,28A$
0020 LET A=10
0030 LET B=10
0040 BBUILD A$,A,B,"AREA",10
0050 PRINT "A=";A,"B=";B
0060 FOR I=1 TO 10 STEP 1
0070 PRINT
0080 NEXT I
0090 PRINT "A$=";A$
0100 END

END OF LISTING


A=  10              B=  10
```



2. Nella stampa prodotta dal programma sottostante si può vedere come i dati stringa vengono ad occupare un numero intero di parole (4 caratteri ognuna) per cui le parole sono riempite con spazi in coda (vedi i due spazi dopo VOLUME).

```
LIST
FILE


0010 DCL 20(A$)
0020 BBUILD A$,"VOLUME",10
0030 PRINT "A$=";A$
0040 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
A$=▥/▩▩VOLUME  ▥▩▩▩▩▩▩▩
```

Gli ultimi 8 caratteri stampati dopo A$= rappresentano il numero 10 in doppia precisione.

Istruzione BEEP

Funzione — Produce una segnalazione acustica.

Formato — **BEEP**

Azione — Il segnalatore acustico emette un suono della durata di 0,2 secondi.

Nota — L'istruzione è particolarmente utile per richiamare l'attenzione dell'operatore su un messaggio visualizzato sul display.

Esempio — La seguente routine richiama l'attenzione dell'operatore sul messaggio: I non supera 98. L'istruzione DELAY impone un ritardo nell'esecuzione dell'istruzione successiva per cui il suono permane e ad ogni esecuzione del ciclo FOR/NEXT riappare il messaggio suddetto.

```
LIST
FILE

0010 FOR I=1 TO 100 STEP 1
0020 BEEP
0030 DISP "I non supera 98"
0040 DELAY 2
0050 DISP
0060 NEXT I
0070 DISP "I e' maggiore di 98"
0080 DELAY 100
0090 END

END OF LISTING
```

Istruzione BPAD

Funzione

Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione, aggiungendo in coda dei caratteri binari.

Formato

BPAD string-var

dove:
string-var
è una variabile stringa di cui si vuole rendere la lunghezza attuale uguale alla lunghezza di allocazione.

Azione

Se la variabile string-var ha un valore il cui numero di caratteri è inferiore alla sua lunghezza di allocazione, in coda ad essa sono aggiunti altrettanti caratteri di riempimento che nella tabella ISO corrispondono al numero decimale 255.

Note

1. L'istruzione BPAD permette la generazione di record di dati aventi la stessa lunghezza, quando si genera un file dati esterno oppure un file su di un supporto di una periferica esterna.

2. Se la variabile specificata nella istruzione ha lunghezza attuale uguale alla sua lunghezza di allocazione, l'istruzione è ignorata.

Esempi

1. Dopo aver creato il file dati esterno ad accesso diretto, di nome BPAD, utilizzando l'istruzione 60 si generano dei record di 60 caratteri.

   In risposta alla istruzione 40 si forniscono le

parole corrispondenti ai numeri ordinali da 1 a 32. Si genera così un file che indica quali sono i caratteri della tabella ISO, dal primo al trentaduesimo.

```
CRE BPAD,(MAT,FDU1),R

LIST
FILE     ISOT

0010 DCL 20(A$),60(B$)
0020 FILES BPAD
0030 FOR I=0 TO 31 STEP 1
0040 INPUT A$
0050 LET B$="Il "+A$+" carattere della Tabella ISO e' "+CHR$(I)
0060 BPAD B$
0070 WRITE :1,B$
0080 NEXT I
0090 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
?
primo
?
secondo
?
terzo
?
quarto
?
```

2. Con il programma sottostante si leggono i record desiderati del file dati generato con l'esempio precedente. Avendo generato dei record tutti della medesima lunghezza la ricerca del dato desiderato risulta facile: ogni dato occupa 16 parole per cui il pointer viene posizionato sulle parole 16*(N-1) +1 dove N è il numero d'ordine del dato desiderato. Prima di stampare il dato si eliminano i caratteri di riempimento con l'istruzione 65 (vedi istruzione DEPAD).

```
LIST
FILE     READIS

0010 DCL 60(A$)
0020 FILES BPAD
0030 DISP "QUALE ELEMENTO ISO VUOI";
0040 INPUT N
0050 SETW :1 TO 16*(N-1)+1
0060 READ :1,A$
0065 DEPAD A$,255
0070 PRINT A$
0080 PRINT
0090 GOTO 30
0100 END

END OF LISTING
RUN
QUALE ELEMENTO ISO VUOI?
1
Il primo carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
2
Il secondo carattere della Tabella ISO e' ┌

QUALE ELEMENTO ISO VUOI?
3
Il terzo carattere della Tabella ISO e' ⊥

QUALE ELEMENTO ISO VUOI?
6
Il sesto carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
30
Il trentesimo carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
32
Il trentaduesimo carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
```

Istruzione BUILD

Funzione

Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa separate con un delimitatore.

Formato

BUILD string-var, {**num-exp** | **string-exp**} [, {**num-exp** | **string-exp**}] ... [; **delimiter**]

dove:

string-var
è una variabile stringa in cui vengono trasferite le stringhe ricavate dai valori delle espressioni specificate

num-exp
è una espressione numerica il cui valore viene trascodificato in una stringa di caratteri ISO e trasferito nella variabile stringa specificata

string-exp
è una espressione stringa il cui valore viene trasferito nella variabile stringa specificata

delimiter
è un numero intero compreso tra zero e 255 il cui carattere che gli corrisponde nella tabella ISO viene interposto, come delimitatore, tra le stringhe ISO che sono trasferite nella variabile stringa specificata.

Azione

Le espressioni sono valutate ed i valori ottenuti sono trascodificati in stringhe di caratteri e quindi trasferiti nell'ordine, da sinistra a destra, nella variabile stringa specificata.

Se è specificato l'operando delimiter, il carattere che gli corrisponde nella tabella ISO viene interposto tra le stringhe suddette nell'ambito della variabile stringa specificata.

Note

1. Il valore corrispondente alla valutazione di una espressione stringa viene trascodificato nella sequenza di caratteri che lo costituisce.

2. Il valore corrispondente alla valutazione di una espressione numerica viene trascodificato in una sequenza di caratteri così composta:

   - il primo carattere è uno spazio oppure il segno meno (-) se il valore suddetto è rispettivamente positivo o negativo

   - i successivi caratteri sono delle cifre decimali che corrispondono al valore suddetto ed assumono la forma di un:

     . numero intero se il valore è un numero intero rappresentabile correttamente con non più di 8 cifre

     . numero decimale in virgola fissa, con 8 cifre significative, se il valore è un numero in valore assoluto compreso tra 0.0999999995 e 99999999.4 (se il numero è minore di 1 viene tralasciato lo zero che precede la parte decimale) o anche se il valore è un numero in valore assoluto minore di 0.0999999995 ma rappresentabile con non più di 8 cifre dopo il punto decimale

     . numero decimale in virgola mobile, con 8 cifre significative, in tutti i rimanenti casi

   - l'ultimo carattere è uno spazio

3. L'istruzione BUILD è utile per trasferire dati dalla memoria principale del P6066 ad una periferica esterna. In questo caso, infatti, i dati numerici devono essere assegnati ad una variabile stringa con una istruzione del tipo: 50 BUILD A$,A,B,C,44 che trasferisce i valori di A,B e C in A$ separati con il carattere virgola. I dati numerici registrati su un supporto di una periferica esterna sono trasferiti in memoria principale ed assegnati ad una variabile stringa. Dalla variabile stringa possono poi essere assegnati ad una variabile numerica eseguendo una istruzione ASSIGN del tipo:
   100 ASSIGN A$ A,B,C,44

con cui tre dati registrati su un supporto esterno e ricevuti in A$ sono assegnati nell'ordine alle variabili A,B e C. Per informazioni sulle istruzioni BASIC che attuano il trasferimento di dati tra P6066 e periferiche esterne si veda il manuale P6066 I/O con periferiche esterne.

Esempi

1. Il seguente programma utilizza l'istruzione BUILD per trasferire nelle variabili D$ e C$ le stringhe di caratteri stampate qui sotto. Si noti che il valore 0.0999999995 è generato nel formato in virgola fissa ma arrotondato a 0.1 perchè le cifre significative sono più di 8. Si noti come il valore -0.0099999999 è generato nel formato in virgola mobile perchè al di fuori del campo definito nella relativa nota qui sopra.

```
FILE     RBUIL1

0010 DCL 30(D$),18(C$)
0020 BUILD D$,0.0999999995,-0.0099999999;44
0030 PRINT "D$=";D$
0040 LET A=100
0050 LET B=10
0060 LET A$="AREA"
0070 LET B$=" DI BASE"
0080 BUILD C$,A$+B$,A*B;44
0090 PRINT "A$=";A$,"B$=";B$,"A$+B$=";A$+B$
0100 PRINT "A*B=";A*B
0110 PRINT "C$=";C$
0120 END

END OF LISTING


D$= .10000000 ,-9.9999999E-03
A$=AREA          B$= DI BASE      A$+B$=AREA DI BASE
A*B= 1000
C$=AREA DI BASE, 1000
```

2. In questo esempio non si è specificato alcun separatore e si può notare come in questo caso le stringhe di caratteri assegnate ad A$ si susseguono consecutivamente.

```
LIST
FILE


0010 DCL 25(A$)
0020 BUILD A$,"AR","EA DI BA","SE DE","L CILI","NDRO"
0030 PRINT "A$=";A$
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A$=AREA DI BASE DEL CILINDRO
```

Istruzione BUILD USING

Funzione

Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa ponendo i caratteri in posizioni predefinite da una istruzione immagine (si veda Istruzione IMMAGINE).

Formato

$$\textbf{BUILD USING} \left\{\begin{matrix}\textbf{line-num}\\ \textbf{string-var}_1\end{matrix}\right\}, \textbf{string-var}_2, \left\{\begin{matrix}\textbf{num-exp}\\ \textbf{string-exp}\end{matrix}\right\} \left[, \left\{\begin{matrix}\textbf{num-exp}\\ \textbf{string-exp}\end{matrix}\right\}\right] \ldots$$

dove:

line-num
è un numero intero che indica il numero di linea di una istruzione IMMAGINE

string-var$_1$
è una variabile stringa il cui contenuto specifica come devono disporsi i caratteri nella variabile stringa

string-var$_2$
è una variabile stringa in cui devono essere trasferiti i valori trascodificati delle espressioni specificate

num-exp
indica una espressione numerica il cui valore deve essere trascodificato in una stringa di caratteri numerici ISO e trasferito, nell'ordine indicato, nella variabile stringa specificata con string-var

string-exp
indica una espressione stringa il cui valore deve essere trasferito, nell'ordine indicato, nella variabile stringa specificata con string-var .

Azione

Le espressioni specificate sono valutate ed i valori ottenuti sono trascodificati in sequenze di caratteri ISO, quindi queste ultime sono trasferite, nell'ordine da sinistra a destra, nella variabile stringa indicata con string-var$_2$. I caratteri, nella variabile stringa

specificata con string-var$_2$, sono disposti come è specificato dai campi della istruzione immagine il cui numero di linea è line-num o del contenuto della variabile specificata con string-var$_1$.

Nota

Si veda l'istruzione IMMAGINE per un completa descrizione di come i caratteri sono disposti in string-var$_2$.

Esempi

1. Il programma sottostante pone nella variabile stringa B$ i valori numerici trascodificati in sequenze di caratteri numerici ISO, posizionati come indicato nell'istruzione 10. Nell'istruzione 50 l'immagine con cui i valori delle rispettive espressioni devono essere posti in C$ è indicata dal contenuto della variabile A$.

```
LIST
FILE

0005 PRINT
0010 : ###   $$$$   ##.###   $$$.###   ###.#↑↑↑↑
0020 DCL 70(A$,B$,C$)
0030 LET A$="'LLLLLLLLLL   LLL   'RRRRRRRRRR   RRR   'CCCCCCCCCC   CCC"
0040 BUILD USING 10,B$,12.7,15.8,-5.75,-5.8,-75.5E-7
0050 BUILD USING A$,C$,"Olivetti","Olivetti","Olivetti"
0060 PRINT "B$=";B$
0065 PRINT
0070 PRINT "C$=";C$
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

B$=  12    $15    -5.750    $-5.800    -75.5E-07

C$=Olivetti      LLL      Olivetti    RRR    Olivetti      CCC
```

2. Nel programma sottostante si mostra cosa accade quando i valori delle espressioni nell'istruzione BUILD USING hanno più caratteri di quelli dei relativi campi nell'immagine.

```
LIST
FILE


0005 PRINT
0010 : ###   $$$$   ##.###   $$$.###   ###.#↑↑↑↑
0020 DCL 70(A$,B$,C$)
0030 LET A$="'LLLLLLLLLL   LLL   'RRRRRRRRRR   RRR   'CCCCCCCCCC   CCC"
0040 BUILD USING 10,B$,12.7,15.8,-5.75,-5.8,-75.5E-7
0050 BUILD USING A$,C$,"Olivetti P6066","Olivetti P6066","Olivetti P6066"
0060 PRINT "B$=";B$
0065 PRINT
0070 PRINT "C$=";C$
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

B$=  12    $15    -5.750    $-5.800    -75.5E-07

C$=Olivetti      LLL      Olivetti    RRR    Olivetti     CCC
```

Istruzione CALL

Funzione

Carica in memoria principale un modulo assembler e ne inizia l'esecuzione.

Formato

CALL string-exp$_1$[,{num-exp / string-exp}]...[;DEB]

dove:

string-exp$_1$
è un'espressione stringa il cui valore indica il nome di un modulo assembler presente in una delle librerie aperte

num-exp
è un'espressione numerica il cui valore è passato come argomento al modulo assembler richiamato

string-exp
è un'espressione stringa il cui valore è passato come argomento al modulo assembler richiamato

DEB
è la stringa omonima e specifica al sistema di caricare il modulo di sistema che permette di eseguire il debugging del modulo assembler caricato in memoria principale

Azione

Il modulo assembler specificato con il valore di string-exp$_1$ è caricato in memoria principale ed eseguito.

Note

1. Il numero di argomenti che si possono passare al modulo assembler specificato è limitato solamente dal numero di caratteri che possono costituire una istruzione BASIC (80 caratteri).

2. Al termine della esecuzione del modulo assembler suddetto viene eseguita l'istruzione esecutiva suc-

cessiva alla CALL.

3. Per ulteriori informazioni sull'impiego dell'istruzione si veda la pubblicazione:"P6066-Ambiente di programmazione in Assembler",codice 3978380 T.

Istruzione CHAIN

Funzione

Termina l'esecuzione di un programma presente in memoria principale e carica in memoria principale il programma specificato, lanciandone l'esecuzione.

Formato

CHAIN |«filename»| |string-var|

dove:

filename
è il nome di un file programma presente in una delle librerie

string-var
è una variabile stringa il cui contenuto è il nome di un file programma presente in una delle librerie

Azione

L'esecuzione del programma residente in memoria principale è terminata, come se fosse eseguita una istruzione END (vedi END). Viene caricato in memoria principale, ed eseguito, il programma con il nome indicato tra apici nell'istruzione o contenuto nella variabile stringa specificata come operando dell'istruzione.

Il programma chiamante è cancellato dalla memoria principale; quindi deve essere registrato in una libreria (comando SAVE) prima che l'istruzione CHAIN sia eseguita.

Note

1. Il programma da caricare in memoria principale è ricercato tra le librerie aperte secondo l'ordine di apertura.

2. La comunicazione di dati tra programmi concatenati può avvenire utilizzando l'istruzione COMMON, vedi COMMON.

3. L'istruzione CHAIN chiude l'accesso da parte del programma utente chiamato ai file dati esterni aperti con il programma chiamante, ma mantiene in memoria principale alcune informazioni relative ai file che prima della esecuzione della istruzione suddetta avevano un numero designatore di file compreso tra 1 e 4. Questo permette al sistema di passare più rapidamente alla esecuzione del programma chiamato se i file dati esterni da esso utilizzati sono quelli suddetti. Il programma chiamato deve specificare i nomi dei file suddetti nella istruzione FILES.

   Quindi per passare più rapidamente l'esecuzione ad un programma chiamato dalla istruzione CHAIN, se abbiamo un programma chiamante come il seguente:

```
  10 FILES A,B,C,D,E,F
     -
     -
     -
 100 FILE: 3,"H"
     -
     -
     -
 200 FILE: 1,"Z"
     -
     -
     -
 300 CHAIN "MAT"
     -
     -
     -
9999 END
```

   il programma chiamato non avrà la seguente struttura:

```
  10 FILES A,B,C,D
     -
     -
     -
 100 FILE: 3,"H"
     -
     -
     -
9999 END
```

   ma bensì la struttura seguente:

```
  10 FILES Z,B,H,D
     -
     -
     -
9999 END
```

Esempio

1. IL programma sottostante, dopo aver registrato su file esterno i primi undici caratteri della tabella ISO, richiama un programma che legge il file dati appena prodotto.

```
LIST
FILE     ISOT

0010 DCL 20(A$),60(B$)
0020 FILES BPAD
0030 FOR I=0 TO 10 STEP 1
0040 INPUT A$
0050 LET B$="Il "+A$+" carattere della Tabella ISO e' "+CHR$(I)
0060 BPAD B$
0070 WRITE :1,B$
0080 NEXT I
0090 CHAIN "READIS"
0100 END

END OF LISTING


RUN
?
primo
?
secondo
?
terzo
?
quarto
?
quinto
?
sesto
?
settimo
?
ottavo
?
nono
?
decimo
?
undicesimo
**** FORMALLY CORRECT PROGRAM ****
QUALE ELEMENTO ISO VUOI?
1
Il primo carattere della Tabella ISO e' ▯

QUALE ELEMENTO ISO VUOI?
11
Il undicesimo carattere della Tabella ISO e' ≡

QUALE ELEMENTO ISO VUOI?
```

Istruzione COMMON

Funzione

Permette di definire delle variabili di programma i cui valori possono essere comunicati tra programmi distinti.

Formato

```
COMMON { num-array ()      }
       { simple-string-var }
       { string-array ()   }
[ { num-array ()      }
  ,{ simple-string-var }
  { string-array ()   } ]...
```

dove:

num-array
è una variabile multipla numerica i cui elementi ricevono (o trasferiscono) dati numerici da (a) un programma BASIC

simple-string-var
è una variabile semplice stringa che riceve (o trasferisce) una stringa di caratteri da (a) un programma BASIC

string-array
è una variabile multipla stringa i cui elementi ricevono (o trasferiscono) stringhe di caratteri da (a) un programma BASIC

Azione

L'istruzione specifica quali sono le variabili che il programma può utilizzare per trasferire valori ad un altro programma o per ricevere valori da un altro programma.

Per poter effettuare tale trasferimento di dati tra programmi il sistema utilizza un'area di memoria, detta area comune, in cui memorizza i valori delle suddette variabili. L'utente può specificare il numero di byte da riservare come area comune introducendo un comando SAVE con l'operando COM=m.

Quando in memoria principale è caricato un programma registrato in una libreria mediante un comando SAVE con l'operando COM=m, il sistema riserva i primi m byte della memoria all'area comune e quindi carica il codice del programma. Così, se il programma che è stato eseguito in precedenza utilizzava un'area comune con lo stesso numero di byte o meno, i valori memorizzati dal programma nell'area suddetta non sono alterati -- e sono disponibili per il programma appena caricato.

Come esempio, supponiamo che sia eseguito il seguente programma che era stato registrato nella libreria MAT con il comando SAV PRO1, (MAT,UD), COM=108

```
 10 COM X(),A$(),Y(),B$
 20 DIM X(2,2),A$(4),Y(5)
 30 DCL SX(),6A$()
 50 X(1,1) = 1
 60 X(1,2) = 2
 70 X(2,1) = 3
 80 X(2,2) = 4
 90 A$(1) = "ROMA"
100 A$(2) = "TORINO"
110 A$(3) = "GENOVA"
120 A$(4) = "MILANO"
130 FOR I=1 TO 5
140 Y(I) = I * 10
150 NEXT I
160 B$ = "OLIVETTI P6066"
          -
          -
          -
1000 CHAIN "PRO2"
9999 END
```

I valori assegnati ai componenti della matrice numerica X(), ai componenti della variabile multipla stringa ad una dimensione A$(), ai componenti del vettore Y() ed alla variabile stringa B$, sono registrati, durante l'esecuzione, a partire dall'inizio della memoria utente, nello stesso ordine con cui le variabili sono specificate nell'istruzione COMMON. La figura 5-1 descrive l'area comune e mostra il numero totale di byte occupato da tale area.

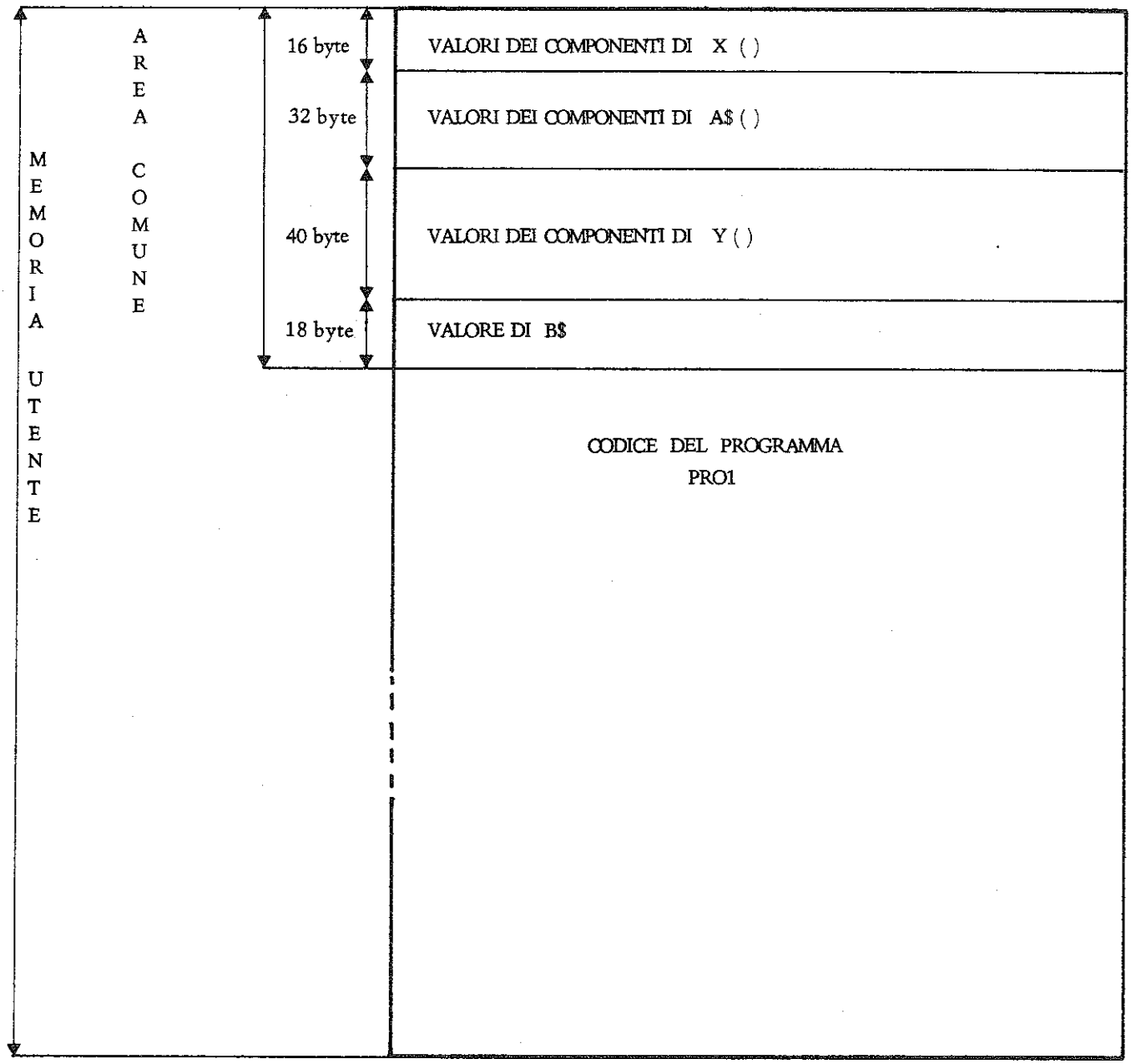


Figura 5-1 L'area comune

Se anche il programma PRO2, caricato in memoria principale con l'istruzione CHAIN di PRO1, è stato registrato in una libreria con la richiesta di un'area comune di 108 byte i valori memorizzati nell'area comune da PRO1 possono essere usati da PRO2, come segue:

```
 10 COMMON X(),A$(),Y(),B$
 20 DIM X(2,2),A$(4),Y(5),N(5),C$(4),C$(4)
 30 DCL S(X(),Z()),6(A$(),C$())
 50 Z(1,1) = X(1,1)
 60 Z(1,2) = X(1,2)
 70 Z(2,1) = X(2,1)
 80 Z(2,2) = X(3,1)
 90 C$(1) = A$(1)
100 C$(2) = A$(2)
110 C$(3) = A$(3)
120 C$(4) = A$(4)
130 FOR I = 1 TO 5
140 N(I) = Y(I)
150 NEXT I
160 D$ = B$
     -
     -
     -
999 END
```

I valori 1, 2, 3 e 4 sono assegnati, rispettivamente, alle variabili Z(1,1), Z(1,2), Z(2,1) e Z(2,2). I valori ROMA, TORINO, GENOVA e MILANO sono assegnati rispettivamente alle variabili C$(1), C$(2), C$(3) e C$(4). I valori 10, 20, 30, 40 e 50 sono assegnati, rispettivamente, alle variabili N(1), N(2), N(3), N(4) ed N(5). Infine, il valore OLIVETTI P6066 è assegnato alla variabile D$. Questi valori sono prelevati dalla area comune, come si può vedere analizzando i programmi PRO1 e PRO2.

Note

1. In un programma BASIC vi può essere una sola istruzione COMMON.

2. Un programma BASIC con una istruzione COMMON non può essere eseguito prima che sia stato registrato in una libreria mediante un comando SAVE con l'operando COM = m.

3. Un programma che utilizza i valori registrati in un'area comune da un programma eseguito in precedenza può essere caricato in memoria principale da una istruzione CHAIN, un comando OLD od un comando RUN filename.

4. Le variabili che si comunicano i valori attraverso un'area comune possono avere nomi differenti.

5. Per richiamare in modo corretto i valori registrati in un'area comune vi deve essere una corretta corrispondenza di locazione del valore rispetto alla variabile che lo utilizza. La posizione di un valore nell'area comune è determinata dalla posizione della corrispondente variabile nell'istruzione COMMON e dal numero di byte occupato dai diversi valori registrati nell'area stessa.

   I valori registrati in un'area comune occupano il seguente numero di byte:

   - ogni elemento di una variabile multipla numerica dichiarata in singola precisione occupa quattro byte

   - ogni elemento di una variabile multipla numerica dichiarata in doppia precisione occupa otto byte

   - ogni elemento di una variabile multipla stringa occupa il numero di byte della sua lunghezza di allocazione, più due byte

   - il valore di una variabile semplice stringa occupa il numero di byte della sua lunghezza di allocazione, più due byte.

   Si noti, infine, che ogni valore numerico è registrato nell'area comune iniziando da una posizione multipla di quattro byte. Così, un valore numerico che è registrato immediatamente dopo una stringa di caratteri può, in alcuni casi, essere registrato dopo tre byte dalla fine della stringa suddetta.

6. I valori numerici registrati nell'area comune come valori di una variabile multipla numerica A possono essere assegnati da un altro programma agli elementi di più variabili multiple numeriche. In questo caso: (1) il numero di elementi di A deve essere uguale alla somma del numero di elementi delle variabili multiple specificate nel secondo programma, (2) il tipo di precisione per la rappresentazione dei valori deve essere lo stesso.

7. Le stringhe di caratteri registrate nell'area comune come valori degli elementi di una variabile multipla stringa A$ possono essere assegnati da un altro programma agli elementi di più variabili multiple stringa oppure a variabili semplici stringa.

Nel primo caso: (1) il numero di elementi di A$ deve essere uguale alla somma del numero di elementi delle variabili multiple stringa specificate nel secondo programma, (2) la lunghezza di allocazione deve essere la stessa.

8. Una stringa di caratteri registrata nell'area comune come valore di una variabile semplice stringa non può essere divisa in sottostringhe da assegnare come valori agli elementi di una o più variabili multiple stringa di un altro programma.

9. Lo spazio richiesto per registrare i valori delle variabili specificate in una istruzione COMMON non può superare il numero di byte specificato per l'area comune dall'operando COM = m del comando SAVE.

10. Lo spazio riservato come area comune per un programma può essere ridotto od allargato quando un altro programma è caricato in memoria principale. Se è ridotto, alcuni valori registrati nell'area comune dal primo programma possono essere perduti. Se è allargato, lo spazio che è aggiunto non conterrà dati significativi finchè non vi sono registrati dal secondo programma.

11. L'utente deve assicurarsi che le variabili usate per comunicare dati tra programmi differenti siano dello stesso tipo.

12. L'area comune non contiene dati significativi finchè non vi siano registrati dal programma.

Istruzione CONVERT

Funzione

Converte ogni carattere di una espressione stringa nel corrispondente codice numerico ISO e viceversa.

Formato

1. **CONVERT string-exp TO num-vector LENGTH num-var**

2. **CONVERT num-vector TO string-var LENGTH num-exp**

dove:

string-exp
è una espressione stringa il cui valore viene convertito, carattere per carattere, nel corrispondente numero intero, in base dieci, della tabella ISO (vedi appendice E)

num-vector
è un vettore numerico ai cui elementi vengono assegnati, ordinatamente, i numeri interi corrispondenti ai caratteri del valore dell'espressione stringa specificata; oppure, nel secondo formato, è un vettore numerico di cui vengono codificati i valori degli elementi nei corrispondenti caratteri ISO

num-var
è una variabile numerica alla quale viene assegnato il numero di caratteri che compongono il valore string-exp

string-var
è una variabile stringa a cui viene assegnata la sequenza di caratteri ISO corrispondente ai valori numerici degli elementi di num-vector

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica quanti elementi di num-vector, iniziando dal primo, devono essere convertiti nei corrispondenti caratteri ISO e quindi assegnati a string-var

Azione

Quando è eseguita l'istruzione indicata con il primo formato, l'espressione string-exp viene valutata ed i caratteri corrispondenti al suo valore vengono convertiti nei corrispondenti numeri in base 10 della tabelle ISO ed assegnati, nell'ordine da sinistra a destra, agli elementi del vettore numerico specificato con num-vector.

All variabile num-var viene assegnato un numero che indica la lunghezza attuale del valore di string-exp.

Quando è eseguita l'istruzione indicata con il secondo formato, vengono convertiti nei corrispondenti caratteri ISO i valori numerici dei primi n elementi del vettore num-vector, arrotondati all'intero più prossimo. La sequenza dei caratteri ISO ottenuta è assegnata alla variabile stringa string-var.

Note

1. Quando è utilizzata l'istruzione CONVERT nel secondo formato, i valori numerici contenuti nel vettore num-vector, arrotondati all'intero più prossimo, devono essere compresi tra zero e 255.

2. Se il numero di caratteri del valore di string-exp è maggiore del numero di componenti del vettore num-vector, nel caso del primo formato, allora viene convertito un numero di caratteri uguale al numero di componenti del vettore suddetto, iniziando dal primo. Viene data una segnalazione di errore recuperabile (appendice D) ed il sistema commuta nello stato di debugging.

3. Se, nel secondo formato, il valore di num-exp arrotondato all'intero più prossimo è maggiore della lunghezza di allocazione della variabile stringa specificata con string-var, allora viene convertito un numero di componenti del vettore num-vector pari al numero di caratteri dichiarati (esplicitamente o implicitamente) per la variabile stringa string-var iniziando dal primo componente. Viene data una segnalazione di errore recuperabile ed il sistema commuta nello stato di debugging.

Esempi

1. Vediamo come si possono convertire le lettere dell'alfabeto nei corrispondenti valori decimali della tabella ISO e viceversa. Come si vede 21.43 è arrotondato a 21.

```
LIST
FILE     CONVER

0010 DCL 21(A$),S(A())
0020 DIM A(21)
0040 CONVERT "ABCDEFGHILMNOPQRSTUVZ" TO A LENGTH B
0050 PRINT
0060 FOR I=1 TO B STEP 1
0070 PRINT "A(";I;")=";A(I),
0080 NEXT I
0090 PRINT
0100 CONVERT A TO A$ LENGTH 21.43
0110 PRINT "A$=";A$
0120 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

A( 1 )= 65      A( 2 )= 66      A( 3 )= 67      A( 4 )= 68      A( 5 )= 69
A( 6 )= 70      A( 7 )= 71      A( 8 )= 72      A( 9 )= 73      A( 10 )= 76
A( 11 )= 77     A( 12 )= 78     A( 13 )= 79     A( 14 )= 80     A( 15 )= 81
A( 16 )= 82     A( 17 )= 83     A( 18 )= 84     A( 19 )= 85     A( 20 )= 86
A( 21 )= 90
A$=ABCDEFGHILMNOPQRSTUVZ
```

2. In questo caso poichè il vettore A ha solo 10 componenti, quando viene eseguita l'istruzione 30, il sistema fornisce una segnalazione di errore recuperabile e si pone nello stato di debugging (la luce del tasto STEP è accesa). Premendo il tasto di console CONTINUE l'esecuzione del programma riprende e vengono convertiti solo i primi 10 caratteri della stringa dell'alfabeto.

```
LIST
FILE     CONVER


0010 DCL 21(A$),S(A())
0030 CONVERT "ABCDEFGHILMNOPQRSTUVZ" TO A LENGTH B
0040 PRINT
0050 FOR I=1 TO 10 STEP 1
0060 PRINT "A(";I;")=";A(I),
0070 NEXT I
0080 PRINT
0090 CONVERT A TO A$ LENGTH B
0100 PRINT "A$=";A$
0110 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
ERROR 8   IN LINE 30

A( 1 )= 65      A( 2 )= 66      A( 3 )= 67      A( 4 )= 68      A( 5 )= 69
A( 6 )= 70      A( 7 )= 71      A( 8 )= 72      A( 9 )= 73      A( 10 )= 76

A$=ABCDEFGHIL
```

Istruzione DATA

Funzione

Crea un file dati interno al programma.

Formato

DATA {**num-constant** | **string-constant**} [, {**num-constant** | **string-constant**}] ...

dove:
num-constant
è una costante numerica
string-constant
è una costante stringa

Azione

Prima dell'esecuzione del programma viene generato, in memoria principale, un file dati interno al programma stesso, composto con tutti i dati numerici e/o stringa specificati con tutte le istruzioni DATA del programma. I dati sono ordinati in sequenza secondo il numero di linea delle istruzioni DATA e, nell'ambito di ogni istruzione DATA, da sinistra a destra. Al file dati interno così prodotto viene associato un pointer che indica quale dato deve essere assegnato di volta in volta alle variabili di programma che compaiono nelle istruzioni READ o MAT READ, quando queste sono eseguite. Quando inizia l'esecuzione del programma, il suddetto pointer indica il primo dato del file interno, che coincide con la prima costante specificata con la prima istruzione DATA, in ordine di numero di linea. I dati contenuti nel file dati interno sono assegnati alle variabili di programma quando sono eseguite le istruzioni READ o MAT READ e dopo ogni assegnazione il pointer viene associato al successivo dato del file suddetto.

Note

1. La istruzione DATA non è una istruzione eseguibile.

2. Le istruzioni DATA si possono collocare in qualun-

que punto del programma.

3. Le costanti stringa specificate in una istruzione DATA sono racchiuse tra virgolette, ma possono essere specificate senza essere racchiuse tra virgolette se i caratteri che le compongono non sono:

   - spazi iniziali e/o finali
   - il carattere virgola (,)

   in questo caso la costante stringa che viene assegnata al file dati interno è la sequenza di caratteri compresa tra il primo e l'ultimo carattere diverso da spazio. In particolare, un numero può essere assegnato ad una variabile stringa specificandolo, nella corrispondente istruzione DATA, racchiuso o non racchiuso tra virgolette (nel secondo caso il numero è rappresentato in memoria principale come una sequenza di caratteri numerici).

4. Il carattere virgolette (") non può far parte di una costante stringa specificata in una istruzione DATA. Per assegnare tale carattere ad una variabile stringa si deve usare l'istruzione RKB (vedi la istruzione omonima).

5. La virgola separa le costanti specificate in una istruzione DATA, per cui è importante non interporla all'interno di una costante, se questa non è racchiusa tra virgolette, altrimenti quest'ultima è assegnata alle variabili di programma divisa in due parti.

6. Se ad una variabile numerica in singola precisione è assegnata una costante numerica nel formato in virgola mobile, con una mantissa con più di 6 cifre significative ed un esponente compreso tra -63 e +63, la mantissa viene troncata ed il sistema non fornisce alcuna segnalazione di errore.

7. Se ad una variabile numerica dichiarata in singola precisione viene assegnato un dato numerico nel formato in virgola mobile con un esponente maggiore di 63, il sistema assegna alla variabile suddetta il valore 9.99999E+63 oppure -9.99999E+63 e dà una segnalazione di errore recuperabile (OVERFLOW)commutando nello stato di debugging (tasto di console STEP acceso).

8. Se ad una variabile numerica dichiarata in singola precisione viene assegnato un valore numerico nel formato in virgola mobile con un esponente minore di -63 il sistema assegna alla variabile suddetta il valore zero e dà una segnalazione di errore recuperabile (UNDERFLOW) commutando nello stato di debugging.

9. Mediante l'istruzione RESTORE si può riassociare il pointer del file dati interno al primo dato, riprendendo così l'assegnazione delle costanti alle variabili di programma dall'inizio del file stesso.

10. Se, dopo aver assegnato ad una variabile di programma l'ultimo dato di un file interno, non si riporta il pointer all'inizio del file mediante l'istruzione RESTORE, l'esecuzione di una successiva istruzione READ o MAT READ produce una segnalazione di errore e l'interruzione della esecuzione del programma. Premendo BREAK il sistema commuta nelllo stato comandi.

11. Ad una variabile numerica non deve essere assegnata una costante stringa.

12. Ad una variabile stringa non deve essere assegnata una costante stringa con più caratteri della sua lunghezza di allocazione.

13. Per ulteriori informazioni si vedano le istruzioni READ, MAT READ e RESTORE.

Esempi

1. Dalla stampa prodotta con l'esecuzione del programma sottostante si può notare come le istruzioni DATA sono considerate ordinate secondo il numero di linea e per esse non hanno alcun effetto le istruzioni di salto (es. GOTO). Se sono presenti in cicli ripetitivi (FOR/NEXT) sono considerate una volta sola. Si noti, infine, come sia possibile specificare delle costanti stringa senza che siano racchiuse tra virgolette.

```
LIST
FILE

0010 DCL 19(H$,G$)
0020 DATA !!!!!,&&&&&,'''''',`````
0030 GOTO 100
0040 DATA Olivetti P6066,1980
0050 FOR I=1 TO 10 STEP 1
0060 PRINT I,
0070 DATA " Manuale Generale "
0080 NEXT I
0090 GOTO 130
0100 DATA 123456789,987654321
0110 DATA Ultima assegnazione
0120 GOTO 50
0130 READ A$,B$,C$,D$,E$,F$,G$,A,B,H$
0140 PRINT "A$=";A$,"B$=";B$,"C$=";C$,"D$=";D$
0150 PRINT "E$=";E$,"F$=";F$,"G$=";G$
0160 PRINT "A=";A,"B=";B
0170 PRINT "H$=";H$
0180 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1                2                3                4                5
 6                7                8                9                10
A$=!!!!!         B$=&&&&&         C$=''''''        D$=`````
E$=Olivetti P6066                 F$=1980          G$= Manuale Generale
A= 1.2345679E+08                  B= 9.8765432E+08
H$=Ultima assegnazione
```

2. Dalla stampa prodotta con l'esecuzione del programma sottostante si nota che:

   - il primo dato numerico viene assegnato alla variabile A, dichiarata in singola precisione, troncando il valore della mantissa alle prime <u>sei</u> cifre significative e non viene data segnalazione di errore perchè l'esponente è compreso tra -63 e +63

   - l'assegnazione dei successivi dati numerici alle relative variabili, dichiarate in singola precisione, provoca ogni volta una segnalazione di errore recuperabile ed il sistema commuta nello stato di debugging (luce di console STEP accesa); premendo il tasto CONTINUE, dopo ogni segnalazione, l'esecuzione del programma riprende ed alle rispettive variabili sono assegnati i valori che si possono vedere (B=9.99999E+63 e D=-9.99999E+63, perchè i valori da assegnare cadono nelle zone di OVERFLOW per la singola precisione; C=Ø ed E=Ø, perchè i valori da assegnare cadono nelle zone di UNDERFLOW per la singola precisione).

```
LIST
FILE


0010 DCL S(A,B,C,D,E)
0020 DATA 123456789E10,123456789E70,1234567899E-90,-123456789E70,-123456789E-90
0030 READ A
0040 READ B
0050 READ C
0060 READ D
0070 READ E
0080 PRINT "A=";A,"B=";B,"C=";C
0090 PRINT "D=";D,"E=";E
0100 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
ERROR 3   IN LINE 40
ERROR 4   IN LINE 50
ERROR 3   IN LINE 60
ERROR 4   IN LINE 70
A= 1.2345600E+18                B= 9.9999900E+63                C= 0
D=-9.9999900E+63                E= 0
```

3. Nel programma sottostante non vi sono sufficienti dati, nel file interno generato dall'istruzione DATA, per soddisfare tutte le assegnazioni dell'istruzione READ per cui viene segnalato un errore e l'esecuzione del programma è sospesa; premendo **BREAK** il sistema commuta nello stato comandi per cui lo si può modificare. Aggiunti i dati al file interno, il programma è eseguito completamente senza provocare segnalazioni di errore.

```
LIST
FILE


0010 DATA A,B,C,D
0020 READ A$,B$,C$,D$,E$,F$,G$
0030 PRINT "A$=";A$,"B$=";B$,"C$=";C$,"D$=";D$
0040 PRINT "E$=";E$,"F$=";F$,"G$=";G$
0050 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
ERROR 88  IN LINE 20
FET 10
0010 DATA A,B,C,D
0010 DATA A,B,C,D,E,F,G
RUN
**** FORMALLY CORRECT PROGRAM ****
A$=A            B$=B            C$=C            D$=D
E$=E            F$=F            G$=G
```

Istruzione DCL

Funzione

Dichiara la lunghezza di allocazione delle variabili stringa e la singola precisione per le variabili numeriche.

Formato

```
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
```

DCL { { S ({ simple-num-var | num-array ( ) } [, { simple-num-var | num-array ( ) }] ...) | n ({ simple-string-var | string-array ( ) } [, { simple-string-var | string-array ( ) }] ...) } SINGLE [, { S ({ simple-num-var | num-array ( ) } [, { simple-num-var | num-array ( ) }] ...) | n ({ simple-string-var | string-array ( ) } [, { simple-string-var | string-array ( ) }] ...) }] ... }

dove:

S
è l'omonimo carattere e specifica che il valore delle variabili numeriche indicate tra parentesi deve essere rappresentato in singola precisione

simple-num-var
è una variabile numerica il cui valore numerico deve essere rappresentato in singola precisione

num-array ( )
indica una variabile multipla (ad una o due dimensioni) numerica i cui elementi (variabili con indice) devono contenere valori numerici rappresentati in singola precisione

n
è un numero intero compreso tra 1 e 1023 che specifica la lunghezza di allocazione delle variabili stringa indicate tra parentesi

simple-string-var
indica una variabile stringa per la quale sono allocati, in memoria principale, n caratteri

string-array ( )
indica una variabile multipla stringa (ad una o due dimensioni) per i cui elementi (variabili con indice) sono allocati n byte in memoria principale

SINGLE
è la parola omonima e specifica che i valori nume-

rici di tutte le variabili numeriche del programma devono essere rappresentati in singola precisione.

Azione

L'istruzione DCL dichiara esplicitamente lo spazio di memoria principale che deve essere riservato alle variabili di programma (vedi appendice F).

L'opzione S, con una o più variabili numeriche tra parentesi, specifica quali variabili numeriche, semplici o con indice, devono avere i relativi valori rappresentati in singola precisione. I valori rappresentati in singola precisione sono elaborati più rapidamente di quelli rappresentati in doppia precisione.

L'opzione n, con una o più variabili stringa tra parentesi, specifica quali variabili stringa, semplici o con indice, potranno avere, come valore, n caratteri.

L'opzione SINGLE specifica che i valori di tutte le variabili numeriche del programma, semplici o con indice, devono essere rappresentati in singola precisione.

Note

1. Se l'opzione S si riferisce ad una sola variabile numerica, semplice o multipla, le parentesi non sono necessarie.

2. Se una opzione n si riferisce ad una sola variabile stringa, semplice o multipla, allora le parentesi non sono necessarie.

3. L'istruzione DCL non è esecutiva.

4. In un programma si possono avere più istruzioni DCL che si riferiscono alla stessa variabile stringa; in questo caso viene assunta la dichiariazione fatta con l'istruzione con numero di linea più alto. In particolare, se per una variabile stringa si fanno più dichiarazioni nell'ambito della stessa istruzione, viene assunta quella più a destra.

5. Se una variabile numerica non compare in alcuna istruzione DCL e nel programma non vi è una istruzione DCL con l'opzione SINGLE, allora i valori della variabile suddetta sono rappresentati in doppia precisione (10 byte sono riservati in memoria principale). Se la variabile numerica suddetta è una variabile multipla, tutti i valori delle variabili con indice che la compongono sono rappresentati in doppia precisione.

6. Se una variabile stringa non compare in alcuna istruzione DCL, allora i valori della variabile suddetta possono essere composti da 16 caratteri al massimo. Se la variabile stringa suddetta è una variabile multipla, tutti i valori delle variabili con indice che la compongono possono essere composti da 16 caratteri al massimo.

7. Il formato DCL SINGLE dichiara la singola precisione solamente per le variabili numeriche, non per le costanti numeriche ed i valori delle funzioni di sistema o i risultati delle espressioni numeriche.

Esempi

1. Nella stampa prodotta dal programma sottostante si vede come i valori delle variabili numeriche sono assunti in singola precisione, infatti i valori ad esse assegnati dalle istruzioni READ sono tutti troncati alla sesta cifra significativa. Alle variabili stringa sono state assegnate sequenze di caratteri con un numero di caratteri eguale al massimo dichiarato nelle relative istruzioni DCL.

```
LIST
FILE     DCL

0010 DCL SA,SB,10A$,20B$
0020 DCL 20[A$(),B$(),C$()]
0030 DCL SA(),SB(),10D$()
0040 READ A,B,A(1),A(2),A(10),B(1,1),B(2,2),B(10,10)
0050 READ A$,B$,C$,A$(1),A$(2),A$(10)
0060 READ B$(1,1),B$(2,2),B$(5,5)
0070 DATA 123456789,987654321,123456789,987654321,123456789,987654321,123456789
0080 DATA 987654321,AAAAAAAAAA,BBBBBBBBBBBBBBBBBBBB,CCCCCCCCCCCCCCCCCCCC
0090 DATA 12345678901234567890,22222222222222222222,ZZZZZZZZZZZZZZZZZZZZ
0100 DATA $$$$$$$$$$$$$$$$$$$$$$,**********************,!!!!!!!!!!!!!!!!!!!!
0120 PRINT "A=";A,"A(1)=";A(1)
0130 PRINT "A(2)=";A(2),"A(10)=";A(10)
0140 PRINT "B(1,1)=";B(1,1),"B(2,2)=";B(2,2)
0150 PRINT "B(10,10)=";B(10,10)
0160 PRINT "A$=";A$,"B$=";B$
0170 PRINT "C$=";C$,"A$(1)=";A$(1)
0180 PRINT "A$(2)=";A$(2),"A$(10)=";A$(10)
```

```
0190 PRINT "B$(1,1)=";B$(1,1),"B$(2,2)=";B$(2,2)
0200 PRINT "B$(5,5)=";B$(5,5)
0210 END

END OF LISTING

RUN
**** FORMALLY CORRECT PROGRAM ****
A= 1.2345600E+08                A(1)= 1.2345600E+08
A(2)= 9.8765400E+08             A(10)= 1.2345600E+08
B(1,1)= 9.8765400E+08           B(2,2)= 1.2345600E+08
B(10,10)= 9.8765400E+08
A$=AAAAAAAAAA   B$=BBBBBBBBBBBBBBBBBBBB
C$=CCCCCCCCCCCCCCCCCCCC         A$(1)=12345678901234567890
A$(2)=22222222222222222222      A$(10)=ZZZZZZZZZZZZZZZZZZZZ
B$(1,1)=$$$$$$$$$$$$$$$$$$$$$$  B$(2,2)=**********************
B$(5,5)=!!!!!!!!!!!!!!!!!!!!!
```

2. Si noti come, se si fanno più dichiarazioni relative ad una stessa variabile, l'ultima dichiarazione è quella che vale.

```
NEW
10 DCL 10 A$
20 DCL 15A$
30 DCL 5B$(),10B$()
40 READ A$,B$(10)
50 DATA ABCDEFGHILMNOPQ,AAAAAAAAAA
60 PRINT "A$=";A$,"B$(10)=";B$(10)
70 END
RUN
**** FORMALLY CORRECT PROGRAM ****
A$=ABCDEFGHILMNOPQ              B$(10)=AAAAAAAAAA
```

3. Nel programma sottostante non esistono istruzioni DCL per cui per le variabili utilizzate valgono le dichiarazioni implicite che prevedono la doppia precisione per la rappresentazione dei valori delle variabili numeriche (il numero stampato è arrotondato alla ottava cifra significativa, come prevede il formato standard di stampa e non alla sesta cifra significativa) e 16 caratteri per i valori delle variabili stringa.

```
NEW
10 READ A,B(1),A$,B$(2)
20 DATA 123456789,987654321,AAAAAAAAAAAAAAAA,BBBBBBBBBBBBBBBB
30 PRINT "A=";A,"B(1)=";B(1)
40 PRINT "A$=";A$,"B$(2)=";B$(2)
50 END
RUN
**** FORMALLY CORRECT PROGRAM ****
A= 1.2345679E+08                B(1)= 9.8765432E+08
A$=AAAAAAAAAAAAAAAA             B$(2)=BBBBBBBBBBBBBBBB
```

Istruzione DEF

Funzione

Definisce una funzione monolinea.

Formato

$$\mathbf{DEF}\left\{\begin{array}{l}\mathbf{FN}\alpha\left[(\left|\begin{array}{l}\textbf{simple-num-var}\\ \textbf{simple-string-var}\end{array}\right|\left[,\left|\begin{array}{l}\textbf{simple-num-var}\\ \textbf{simple-string-var}\end{array}\right|\right]\ \ldots)\right]=\textbf{num-exp}\\ \mathbf{FN}\alpha\,\$\left[(\left|\begin{array}{l}\textbf{simple-num-var}\\ \textbf{simple-string-var}\end{array}\right|\left[,\left|\begin{array}{l}\textbf{simple-num-var}\\ \textbf{simple-string-var}\end{array}\right|\right]\ \ldots)\right]=\textbf{string-exp}\end{array}\right\}$$

dove:

FNα

indica il nome di una funzione monolinea di tipo numerico; α è sostituito da una lettera maiuscola dell'alfabeto inglese

simple-num-var

indica una variabile semplice numerica, definita come parametro della funzione; ad essa sono assegnati i valori specificati nel richiamo di funzione

simple-string-var

indica una variabile semplice stringa, definita come parametro della funzione; ad essa sono assegnati i valori specificati nel richiamo di funzione

num-exp

è una espressione numerica che definisce l'algoritmo della funzione numerica

FNα\$

indica il nome di una funzione monolinea di tipo stringa; α è sostituito con una lettera maiuscola dell'alfabeto inglese

string-exp

è una espressione stringa che definisce l'algoritmo della funzione stringa

Azione

Il formato con FNα definisce una funzione monolinea di tipo numerico.

La parte a sinistra del segno uguale specifica il nome della funzione ed a quali variabili, dette parametri, sono assegnati i valori forniti dagli argomenti. La funzione viene infatti richiamata, in una qualunque istruzione di programma in cui possono apparire delle espressioni numeriche, con il suo nome seguito eventualmente da costanti, variabili ed espressioni, i cui valori sono detti argomenti della funzione e sono assegnati nell'ordine, da sinistra a destra, ai parametri della funzione. Gli argomenti devono essere compresi tra parentesi e separati con una virgola. I parametri sono distinti dalle variabili del programma, per cui si possono avere variabili di programma con lo stesso nome delle variabili che sono utilizzate come parametri di una funzione. La parte a destra del segno uguale è una espressione numerica che può contenere come operandi, oltre alle variabili definite come parametri, altre variabili che sono utilizzate nel programma e per tale motivo sono dette variabili globali. Come operandi di detta espressione vi possono essere anche dei richiami di funzione di sistema o definite dall'utente (monolinea o multilinea). Quando viene eseguita una istruzione di programma che contiene il richiamo alla funzione suddetta, alle variabili definite come parametri sono assegnati i valori posseduti in quel momento dagli argomenti e alle variabili globali quelli che esse hanno in quel momento; l'espressione è eseguita ed il valore numerico ottenuto è utilizzato dalla istruzione di programma suddetta.

Il formato con FNα$ definisce una funzione monolinea di tipo stringa. Per essa valgono le stesse definizioni (parametri, argomenti) e considerazioni del punto precedente. In questo caso il risultato ottenuto dalla valutazione dell'espressione a destra del segno uguale è una stringa di caratteri.

Note

1. La funzione FNα$ può essere richiamata in qualunque istruzione di programma in cui può apparire una espressione stringa.

2. L'istruzione DEF non è eseguibile e può essere posta in qualunque punto di un programma, anche dopo l'istruzione in cui è richiamata.

3. In un programma si possono definire 26 funzioni monolinea per ciascun tipo (numerico o stringa) e per ogni definizione si possono utilizzare parametri a-

venti lo stesso nome.

4. Il numero di parametri di una funzione monolinea (numerica o stringa) non può essere maggiore di 15.

5. I valori delle variabili numeriche specificate nell'istruzione DEF come parametri sono rappresentati in doppia precisione, anche se le variabili numeriche, specificate nel richiamo di funzione come argomenti, sono state dichiarate in singola precisione.

6. Le variabili stringa, specificate nell'istruzione DEF come parametri, hanno una lunghezza di allocazione pari a quella dei relativi argomenti.

7. Non si possono definire in uno stesso programma due funzioni monolinea con lo stesso nome.

8. Se vi sono delle segnalazioni di errore riferite ad una funzione durante l'esecuzione di un programma, queste fanno riferimento al numero di linea della istruzione contenente il richiamo di funzione stesso.

9. E' bene non fare delle definizioni ricorsive come, per esempio:

```
        DEF FNA = FNA        (è proibita)
oppure  DEF FNA = X+FNB }    (non è proibita ma è
   con  DEF FNB = Y+FNA }    sconsigliabile)
```

10. Gli argomenti specificati nel richiamo di una funzione monolinea devono corrispondere in numero, ordine e tipo, ai parametri specificati nella relativa definizione di funzione.

11. Si noti che lo spazio di memoria principale, occupato dalla variabile usata come parametro, è rilasciato al termine della esecuzione della funzione.

12. Il valore di ritorno di una funzione monolinea di tipo stringa non può contenere più di 16 caratteri.

Esempi

1. Nel programma sottostante si richiama due volte una funzione monolinea numerica, la seconda volta per utilizzare il valore di ritorno di un'altra espressione numerica.

```
LIST
FILE


0010 DEF FNA(A,B,C)=A*B*C+F
0020 LET F=10
0030 LET A=2
0040 LET B=3
0050 LET C=5
0060 LET D=100
0070 LET F=A*B*C
0080 LET Z=9
0090 LET Y=6
0100 LET F=150
0110 PRINT "FNA=";FNA(4,5*5,SQR(D)),"FNA+B=";FNA(Z,Z+A,Y)+B
0120 END

END OF LISTING


RUN
FNA= 1150        FNA+B= 747
```

2. Nel programma sottostante si definisce una funzione monolinea numerica che utilizza come purametri delle variabili stringa.

```
LIST
FILE


0010 DEF FNA(A$,B$)=SCN(A$,B$,1,1)
0020 LET B$="AAABBB"
0030 LET C$="B"
0040 LET Z=FNA(B$,C$)
0050 PRINT "Z=";Z
0060 END

END OF LISTING


RUN
Z= 4
```

3. Ecco un altro esempio di definizione di funzione monolinea numerica con variabili stringa come parametri.

```
NEW
10 DEFFNB(A$,B$,A)=LEN(A$)+LEN(B$)+A
20 D$="DDDDDDDDDD"
30 F$="FFFFF"
40 X=100
50 PRINT "FNB(D$,F$,X)=";FNB(D$,F$,X)
60 END
RUN
**** FORMALLY CORRECT PROGRAM ****
FNB(D$,F$,X)= 115
```

4. Nel programma sottostante viene definita ed utilizzata una funzione monolinea di tipo stringa.

```
LIST
FILE


0010 DEF FNA$(A$,A,B,B$,C$,D$,C,D)=EXT$(A$,A,B)+REP$(B$,C$,D$,C,D)
0020 LET Q$="AREA VOLUME PESO"
0030 LET I=6
0040 LET F=11
0050 LET S$="DILLASFIRA"
0060 LET X$="I"
0070 LET Y$="E"
0080 PRINT "FNA$=";FNA$(A$,I,F,S$,X$,Y$,2,1)
0090 END

END OF LISTING





RUN
**** FORMALLY CORRECT PROGRAM ****
FNA$=VOLUMEDELLASFERA
```

Istruzioni DEF/FNEND

Funzione — Definiscono una funzione multilinea.

Formato

```
DEF {FNα  } [( {simple-num-var   } [, {simple-num-var   }] ...)] [{simple-num-var   } [, {simple-num-var   }] ...]
    {FNα$ }    {simple-string-var}    {simple-string-var}          {simple-string-var}      {simple-string-var}
[Istruzione BASIC]
[LET] {FN*  = num-exp   }
      {FN*$ = string-exp}
[Istruzione BASIC]
[[LET] {FN*  = num-exp   }]
       {FN*$ = string-exp}
[Istruzione BASIC]
FNEND
```

dove:

FNα
indica il nome di una funzione multilinea di tipo numerico; α è sostituito con una lettera maiuscola dell'alfabeto inglese

FNα$
indica il nome di una funzione multilinea di tipo stringa; α è sostituito con una lettera maiuscola dell'alfabeto inglese

simple-num-var (compresa tra parentesi tonde)
indica una variabile numerica semplice, definita come parametro della funzione (numerica o di tipo stringa)

simple-string-var (compresa tra parentesi tonde)
indica una variabile stringa semplice, definita come parametro della funzione (numerica o di tipo stringa)

simple-num-var (non compresa tra parentesi tonde)
indica una variabile numerica semplice, definita come variabile locale della funzione (numerica o di tipo stringa)

simple-string-var (non compresa tra parentesi tonde)
indica una variabile stringa semplice, definita come variabile locale della funzione (numerica o di tipo stringa)

FN*
è una variabile numerica (pseudovariabile) non utilizzabile nel programma al di fuori della definizione di funzione multipla numerica, alla quale viene assegnato il valore della espressione numerica specificata alla destra del segno uguale

FN*$
è una variabile stringa (pseudovariabile) non utilizzabile nel programma, al di fuori della definizione di funzione monolinea del tipo stringa, alla quale viene assegnato il valore dell'espressione stringa specificata alla destra del segno uguale

num-exp
indica una espressione numerica, specificata nello ambito di una definizione di funzione multilinea numerica, il cui valore è assegnato alla pseudovariabile numerica FN* specificata alla sinistra del segno uguale

string-exp
indica una espressione stringa, specificata nello ambito di una definizione di funzione multilinea di tipo stringa, il cui valore è assegnato alla pseudovariabile stringa FN*$

FNEND
è una parola chiave BASIC (vedi istruzione FNEND). Specifica che l'istruzione precedente, in ordine di numero di linea, è l'ultima istruzione della definizione di funzione multilinea (numerica o di tipo stringa).

Azione

Il formato con FNα definisce una funzione multilinea di tipo numerico. Le istruzioni BASIC, comprese tra l'istruzione FNα e l'istruzione FNEND, sono eseguite quando in una istruzione del programma compare il richiamo di funzione corrispondente. Il richiamo di funzione è costituito dal nome della funzione, seguito, eventualmente, da una o più costanti, variabili o espressioni, comprese tra parentesi; i cui valori sono detti argomenti della funzione. I valori degli argomenti sono assegnati nell'ordine, da sinistra a destra, alle variabili specificate nella definizione di funzione come parametri. La prima istruzione della definizione di funzione specifica, oltre al nome della

funzione stessa, le variabili che sono utilizzate dalla funzione come parametri ed eventuali variabili locali alle quali sono assegnati i valori con istruzioni contenute nell'ambito della definizione di funzione. Nell'ambito della definizione di funzione vi possono essere una o più istruzioni di assegnazione del valore di una espressione numerica ad una pseudovariabile numerica FN*.

Tutte le istruzioni della definizione di funzione possono contenere variabili definite come parametri, variabili locali e variabili specificate in altre istruzioni del programma (variabili globali). Quando la funzione è richiamata, le variabili definite come parametri sono inizializzate con il valore dei rispettivi argomenti, le variabili locali non sono inizializzate e le variabili globali assumono l'ultimo valore ad esse assegnato nell'ambito del programma.

Il valore numerico restituito dalla funzione è quello assegnato ad FN* prima dell'esecuzione dell'istruzione FNEND.

Il formato in cui compare FNα$ definisce una funzione multilinea di tipo stringa. Valgono per questo formato tutte le considerazioni suddette ma si deve osservare che in questo caso la pseudovariabile a cui viene assegnato un valore è di tipo stringa (FN*$) e l'espressione relativa, a destra del segno uguale, è anch'essa di tipo stringa.

Note

1. Se, durante l'esecuzione di un programma, è incontrata una definizione di funzione, il controllo della esecuzione passa direttamente alla prima istruzione successiva alla istruzione FNEND.

2. Il richiamo di una funzione multilinea può essere utilizzato in tutte le istruzioni di programma in cui può apparire una espressione.

3. Una definizione di funzione multilinea può essere presente in qualunque punto di un programma.

4. In una definizione di funzione multilinea, la somma delle variabili specificate come parametri e delle variabili locali non può essere maggiore di 15.

5. E' importante notare che lo spazio di memoria principale occupato dai parametri e dalle variabili locali è rilasciato al termine della esecuzione della funzione.

6. I parametri e le variabili locali sono distinti dalle variabili globali aventi lo stesso nome. Non si possono avere parametri e variabili locali con lo stesso nome.

7. In un programma si possono definire 26 funzioni multilinea di tipo numerico e 26 funzioni multilinea di tipo stringa. Per ogni definizione di funzione si possono utilizzare variabili locali e parametri aventi lo stesso nome.

8. Alle variabili locali si deve assegnare un valore prima del loro impiego, altrimenti, quando sono eseguite istruzioni che le utilizzano, viene fornita una segnalazione di errore recuperabile ed il sistema commuta nello stato di debugging in attesa di una decisione da parte dell'utente (il sistema assegna implicitamente alle variabili locali numeriche il valore zero ed alle variabili locali stringa il valore stringa nulla).

9. Il valore di una variabile specificata come parametro può essere modificato all'interno della funzione.

10. Le variabili definite come parametri devono corrispondere in numero, ordine e tipo agli argomenti specificati nel relativo richiamo di funzione.

11. Il valore di una variabile globale può essere modificato durante l'esecuzione di una funzione ed il nuovo valore viene mantenuto dalla variabile quando il controllo dell'esecuzione ritorna al programma chiamante.

12. Le variabili numeriche definite come parametri sono rappresentate in doppia precisione. Le variabili stringa definite come parametri assumono la stessa lunghezza di allocazione degli argomenti.

13. I valori delle variabili locali numeriche sono rappresentati in doppia precisione. Si può definire la lunghezza di allocazione di una variabile locale

di tipo stringa mediante una istruzione DCL nello ambito della definizione di funzione.

14. I valori delle variabili globali numeriche sono rappresentati con la stessa precisione definita per esse nel programma principale. Le variabili globali di tipo stringa hanno la stessa lunghezza di allocazione definita per esse nel programma principale. Non si possono modificare il tipo di rappresentazione delle variabili globali numeriche e la lunghezza di allocazione delle variabili globali di tipo stringa, utilizzando un'istruzione DCL all'interno di una definizione di funzione multilinea.

15. Nell'ambito di una definizione di funzione multilinea, si può definire la lunghezza di allocazione della pseudovariabile FN*$ con un'apposita istruzione DCL.

16. Non si possono utilizzare alla destra del segno uguale la pseudovariabile FN* ed FN*$.

17. In tutte le istruzioni di una definizione di funzione multilinea in cui può apparire una espressione, possono apparire dei richiami di funzione di sistema o definite dall'utente (monolinea o multilinea).

18. Le segnalazioni d'errore riguardanti l'esecuzione di una funzione multilinea sono riferite al numero di linea dell'istruzione in cui compare il relativo richiamo di funzione. Le segnalazioni di errore segnalate durante la preesecuzione di una funzione multilinea sono riferite alla istruzione FNEND.

19. Non si può definire un'altra funzione multilinea all'interno di una definizione di funzione multilinea.

20. In una definizione di funzione multilinea deve esserci almeno una istruzione di assegnazione di valore alla relativa pseudovariabile (FN* se la funzione è numerica od FN*$ se la funzione è di tipo stringa).

21. Una istruzione esterna od una definizione di funzione multilinea non può passare il controllo del-

l'esecuzione del programma ad una istruzione interna a tale definizione.

22. Una istruzione interna ad una definizione di funzione multilinea non può passare il controllo della esecuzione del programma ad una istruzione esterna a tale definizione.

23. All'interno di una definizione di funzione non vi può essere una istruzione STOP.

24. In un programma una funzione multilinea con un dato nome può essere definita una sola volta.

25. Una funzione multilinea può richiamare se stessa.

Esempi

1. Nel programma sottostante viene riportata una definizione di funzione multilinea numerica. I parametri sono A,B e C. Le variabili locali sono D,E ed F. Si utilizzano anche le variabili globali H e C$. Gli argomenti sono 5,6 ed 8. Si dimostra come il valore ritornato da FN* è quello calcolato nella ultima istruzione di assegnazione relativa eseguita prima di FNEND. In un caso tale valore è calcolato con la prima espressione riferita ad FN*, nell'altro caso con la seconda espressione riferita ad FN*. Per terminare l'esecuzione di questo programma si deve premere il tasto di console BREAK.

```
LIST
FILE

0010 DEF FNA(A,B,C)D,E,F
0020 DISP "INTRODUCI D";
0030 INPUT D
0040 DISP "INTRODUCI E";
0050 INPUT E
0060 DISP "INTRODUCI F";
0070 INPUT F
0080 DISP "SALTO";
0090 INPUT C$
0100 IF C$="SI" THEN 140
0110 PRINT "NON HO SALTATO L'ISTRUZIONE 120"
```

```
0120 LET FN*=(A+B+C+D+E+F)/H
0130 GOTO 160
0140 PRINT "HO SALTATO L'ISTRUZIONE 120"
0150 LET FN*=A+D+LEN(A$)
0160 FNEND
0170 LET H=100
0180 LET A$="Olivetti P6066"
0190 PRINT "FNA=";FNA(5,6,8)
0200 DISP "VUOI RIPETERE IL PROGRAMMA";
0210 INPUT B$
0220 IF B$="SI" THEN 170
0230 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
INTRODUCI D?
1
INTRODUCI E?
2
INTRODUCI F?
3
SALTO?
NO
NON HO SALTATO L'ISTRUZIONE 120
FNA= .25
VUOI RIPETERE IL PROGRAMMA?
SI
INTRODUCI D?
1
INTRODUCI E?
2
INTRODUCI F?
3
SALTO?
SI
HO SALTATO L'ISTRUZIONE 120
FNA= 20
VUOI RIPETERE IL PROGRAMMA?
SI
INTRODUCI D?
20
INTRODUCI E?
30
INTRODUCI F?
40
SALTO?
NO
NON HO SALTATO L'ISTRUZIONE 120
FNA= 1.09
VUOI RIPETERE IL PROGRAMMA?
NO
```

2. Nel programma sottostante si può vedere che se si hanno variabili locali e variabili globali con lo stesso nome, il valore utilizzato dalle istruzioni della funzione multilinea e quello corrispondente alle variabili locali. Quando è eseguita l'istruzione 65, il sistema commuta nello stato di debugging. Premendo B ed END OF LINE sul display appare il valore della variabile globale B.

```
LIST
FILE


0010 DEF FNA B
0020 LET B=10
0030 PRINT "B=";B
0040 LET FN*=B
0050 FNEND
0055 LET B=15
0060 PRINT "FNA=",FNA
0065 STOP
0070 END

END OF LISTING



RUN
B= 10
FNA= 10
STOP       IN LINE 65
B
 15
```

3. Nel programma sottostante si vede come un parametro ed una variabile globale possono avere lo stesso nome. E' l'argomento B che fornisce il valore al parametro A.

```
LIST
FILE


0010 DEF FNZ(A)
0020 PRINT "A=";A
0030 LET FN*=2*A
0040 FNEND
0050 LET A=115
0060 LET B=5
0070 PRINT "FNZ=";FNZ(B)
0080 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A= 5
FNZ= 10
```

4. Nel programma seguente viene utilizzata una funzione multilinea di tipo stringa.

```
LIST
FILE     WWW


0010 PRINT "SE RISPONDI NO LA FUNZIONE REGISTRA I CODICI ISO"
0020 DCL 256 G$,80(F$,H$,I$,X$,K$,L$,Z$,U$),1600$,500M$
0030 LET F$="Il Sistema P6066 puo` funzionare"
0040 LET H$=" in diversi modi che assumono il nome di Stati."
0050 LET I$=" Lo Stato Comandi e` il modo di funzionamento che"
0060 LET X$=" permette di introdurre ed eseguire tutti i comandi di sistema."
0070 LET K$="Lo Stato di Debugging e` il modo di funzionamento che per"
0080 LET L$="mette l'impiego degli strumenti di verifica dei programmi."
00902$="Lo Stato Calcoli Immediati e` il modo di funzionamento del Sistema in c"
0100 LET U$="ui si possono eseguire immediatamente delle espressioni numeriche."
0110 LET O$=Z$+U$
0120 PRINT FNA$("FINE"," DI")
0130 DEF FNA$(A$,B$)C$
0140 DISP "VUOI MODIFICARE UN TESTO";
0150 INPUT S$
0160 IF S$="SI" THEN 340
0170 FILES CODISO
0180 SCRATCH :1
0190 FOR I=0 TO 255 STEP 1
0200 WRITE :1,CHR$(I)
0210 NEXT I
0220 DISP "REGISTRATI I CODICI ISO"
0230 DELAY 60
0240 LET C$=" CODICI"
0250 LET FN*$=A$+B$+C$
0260 RESTORE :1
0270 LET G$=""
0280 FOR I=0 TO 255 STEP 1
0290 READ :1,D$
0300 LET G$=G$+D$
0310 NEXT I
0320 PRINT G$
0330 GOTO 450
0340 DISP "MODIFICO UN TESTO"
0350 LET C$=" TESTO"
0360 DELAY 100
0370 LET FN*$=A$+B$+C$
0380 LET M$=F$+H$+I$+X$+K$+L$
0390 PRINT "Il vecchio testo e`:"
0400 PRINT M$
0410 PRINT
0420 PRINT
0430 PRINT "Il nuovo testo e`:"
0440 PRINT REP$(M$,EXT$(M$,81,191),O$,1,1)
0450 FNEND
0460 END

END OF LISTING



RUN
SE RISPONDI NO LA FUNZIONE REGISTRA I CODICI ISO
VUOI MODIFICARE UN TESTO?
SI
MODIFICO UN TESTO
Il vecchio testo e`:
Il Sistema P6066 puo` funzionare in diversi modi che assumono il nome di Stati.
Lo Stato Comandi e` il modo di funzionamento che permette di introdurre ed esegu
ire tutti i comandi di sistema.Lo Stato di Debugging e` il modo di funzionamento
 che permette l'impiego degli strumenti di verifica dei programmi.


Il nuovo testo e`:
Il Sistema P6066 puo` funzionare in diversi modi che assumono il nome di Stati.
Lo Stato Calcoli Immediati e` il modo di funzionamento del Sistema in cui si pos
sono eseguire immediatamente delle espressioni numeriche.Lo Stato di Debugging e
` il modo di funzionamento che permette l'impiego degli strumenti di verifica de
i programmi.
FINE DI TESTO
```

```
RUN
SE RISPONDI NO LA FUNZIONE REGISTRA I CODICI ISO
VUOI MODIFICARE UN TESTO?
NO
REGISTRATI I CODICI ISO
■┌┴┘└┤▒│°└┐├┬┼└▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]↑_`abcdefghijklmnopqrstuvwxyz{|}~▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
```

Istruzione DELAY

Funzione

Ritarda l'esecuzione dell'istruzione successiva.

Formato

DELAY time

dove:
time
è un numero intero positivo che indica di quanti decimi di secondo deve essere ritardata l'esecuzione dell'istruzione successiva.

Azione

L'istruzione successiva è eseguita dopo tanti decimi di secondo quanti sono specificati con il numero specificato con time.

Note

1. Premendo il tasto di console CONTINUE l'istruzione di programma successiva all'istruzione DELAY viene eseguita immediatamente.

2. L'istruzione DELAY permette l'impiego di successive istruzioni DISP poichè, posta dopo ogni istruzione DISP, permette la lettura del relativo messaggio sul display.

3. L'istruzione DELAY, utilizzata dopo una istruzione DISP, permette la lettura di messaggi con più di 32 caratteri. In questo caso, poichè il messaggio rimane sul display per il tempo programmato, si può, premendo il tasto [→], spostare il testo del messaggio sul display verso sinistra.

   Esempio: La routine sottostante mostra l'impiego dell'istruzione DELAY. Il messaggio prodotto dall'istruzione 10 si può leggere, perchè rimane sul display per 10 secondi.

Premendo il tasto [→] si può spostare verso sinistra il testo sul display.

```
LIST
FILE     DELAY

0010 DISP "Tra dieci secondi vedrai un altro messaggio"
0020 DELAY 100
0030 DISP "Questa routine mostra"
0040 DELAY 20
0050 DISP "l'impiego della istruzione"
0060 DELAY 20
0070 DISP "DELAY"
0080 DELAY 20
0090 END

END OF LISTING


RUN
Tra dieci secondi vedrai un altro messaggio
Questa routine mostra
l'impiego della istruzione
DELAY
```

Istruzione DEPAD

Funzione

Rimuove in una variabile stringa i caratteri di riempimento specificati.

Formato

DEPAD string-var, n

dove:

string-var

è una variabile stringa di cui si vuole modificare il valore, eliminando i caratteri di riempimento specificati con n

n

è un numero intero, compreso tra Ø e 255, che specifica quali sono i caratteri da togliere al valore della variabile specificata con string-var.

Azione

Nella variabile stringa sono rimossi i caratteri corrispondenti nella tabella ISO al numero specificato con n, se detti caratteri sono in coda nella stringa che costituisce il valore della variabile suddetta.

Esempi

1. Vediamo dall'esecuzione del seguente programma come agisce l'istruzione DEPAD. Si noti come in A$ sono tolti gli asterischi in coda e non quelli all'inizio della stringa che ne costituisce il valore.

```
LIST
FILE

0010 DCL 70 (A$,B$,C$),150 D$
0020 LET A$="*** Il sistema P6066 puo` essere **********"
0030 LET B$="utilizzato per eseguire ######"
0040 LET C$="immediatamente dei calcoli algebrici***$$$$$"
0050 LET D$=A$+B$+C$
0060 PRINT "In D$ c'e` il testo:"
0070 PRINT D$
0080 DEPAD A$,42
0090 DEPAD B$,35
```

```
0100 DEPAD C$,36
0110 PRINT
0120 PRINT "Ora in D$ c'e` il testo:"
0130 LET D$=A$+B$+C$
0140 PRINT D$
0150 PRINT
0160 PRINT "Questo perche` in A$,B$ e C$ sono stati tolti i caratteri in coda,"
0170 PRINT "infatti ora i rispettivi contenuti sono:"
0180 PRINT "A$=";A$
0190 PRINT "B$=";B$
0200 PRINT "C$=";C$
0210 END

END OF LISTING


·RUN
In D$ c'e` il testo:
*** Il sistema P6066 puo` essere **********utilizzato per eseguire ######immedia
tamente dei calcoli algebrici***$$$$$

Ora in D$ c'e` il testo:
*** Il sistema P6066 puo` essere utilizzato per eseguire immediatamente dei calc
oli algebrici***

Questo perche` in A$,B$ e C$ sono stati tolti i caratteri in coda,
infatti ora i rispettivi contenuti sono:
A$=*** Il sistema P6066 puo` essere
B$=utilizzato per eseguire
C$=immediatamente dei calcoli algebrici***
```

2. L'esecuzione dell'istruzione DEPAD non modifica, in questo caso, il contenuto di A$. Come si vede, infatti, in coda al contenuto di A$ non esiste il carattere asterisco.

```
LIST
FILE


0010 DCL 23 A$
0020 LET A$="## Olivetti ** P6066 ##"
0030 DEPAD A$,42
0040 PRINT "A$=";A$
0050 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
A$=## Olivetti ** P6066 ##
```

Istruzione DIM

Funzione

Specifica le dimensioni delle variabili multiple di un programma.

Formato

DIM array (rows [, columns]) [, array (rows [, columns])]...

dove:

array
è il nome di una variabile multipla numerica o di tipo stringa

rows
è un numero intero positivo che specifica quante sono le componenti (variabili con indice) della variabile multipla ad una dimensione di nome array

columns
è un numero intero positivo che, insieme al numero specificato con rows, indica quante sono le componenti (variabili con indice) della variabile multipla a due dimensioni di nome array.

Azione

Quando, dentro le parentesi che seguono il nome di una variabile multipla, è specificato un solo numero, la variabile ha una sola dimensione ed è composta da un numero di variabili con indice pari a row.

Quando, dentro le parentesi che seguono il nome di una variabile multipla, sono specificati due numeri separati da una virgola, la variabile ha due dimensioni ed è composta da un numero di variabili con indice pari al prodotto rows * columns.

Note

1. L'istruzione DIM è non esecutiva, quindi può essere specificata in qualunque posizione nell'ambito di un programma.

2. In un programma vi può essere più di una istruzione DIM.

3. In un programma, non vi possono essere variabili multiple ad una dimensione e variabili multiple a due dimensioni, con lo stesso nome.

4. Se in un programma vi sono più istruzioni DIM che si riferiscono alle stesse variabili multiple, viene assunta la dichiarazione relativa all'istruzione DIM con numero di linea più alto.

5. Se in una istruzione DIM compare più volte la stessa variabile multipla, viene assunta la dichiarazione più a destra nell'ambito della istruzione.

6. Una variabile multipla ad una dimensione, che non compare in alcuna istruzione-DIM, ha 10 componenti. Le relative variabili con indice potranno avere un indice variabile da 1 a 10.

7. Una variabile multipla numerica a due dimensioni, che non compare in alcuna istruzione DIM, ha 10*10 componenti.

8. Una variabile multipla stringa a due dimensioni, che non compare in alcuna istruzione DIM, ha 5x5 componenti.

9. Per una variabile multipla a due dimensioni, non si possono dichiarare dimensioni il cui prodotto superi 65535.

10. Alcune istruzioni (vedi le istruzioni MAT alla fine del capitolo) permettono di modificare il numero di righe e di colonne di una matrice numerica, purchè il loro prodotto non superi il prodotto delle dimensioni specificate nella istruzione DIM (oppure 10*10, se la matrice non è specificata in una istruzione DIM).

Esempi

1. La routine sottostante mostra come vengono dichiarate le dimensioni di due variabili multiple numeriche.

```
LIS
FILE


0010 DIM A(20),B(20,15)
0020 LET A(1)=A(20)=100
0030 LET B(1,1)=B(20,15)=12
0040 PRINT A(1),A(20),B(1,1),B(20,15)
0050 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
 100              100              12               12
```

2. Vediamo, nella routine sottostante, come nel caso di più dichiarazioni con le istruzioni DIM, quella che vale è l'ultima.

```
FILE


0010 DIM A(20),A(30)
0020 DIM B(12,13)
0030 DIM B(15,14)
0040 LET A(30)=30
0050 LET B(15,14)=1514
0060 PRINT A(30),B(15,14)
0070 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
 30               1514
```

3. Nella routine sottostante si vede che l'indice di una variabile con indice può essere maggiore di quello specificato nella istruzione DIM, se una precedente istruzione MAT INPUT modifica le dimensioni della matrice.

```
LIST
FILE


0010 DIM A(5,4)
0020 MAT INPUT A(2,10)
0030 PRINT "A(2,10)=";A(2,10)
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
?
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
A(2,10)= 20
```

Istruzione DISP

Funzione

Visualizza dati e testi, sul display, in formato standard.

Formato

dove:

num-exp
indica una espressione numerica il cui valore deve essere visualizzato sul display

string-exp
indica una espressione stringa il cui valore deve essere visualizzato sul display

TAB (num-exp)
è una funzione di sistema che indica la posizione, corrispondente al valore di num-exp arrotondato all'intero più prossimo, indirizzata dal pointer del buffer di display

,
indica che il pointer del buffer di display deve indirizzare la successiva zona di display

;
indica che il pointer del buffer di display non deve modificare l'indirizzo

Azione

I valori delle espressioni specificate nella istruzione sono visualizzati sul display, nell'ordine con cui compaiono nella istruzione, come descritto nelle note successive.

La posizione sul display dei caratteri che sono visualizzati è controllata mediante l'impiego della virgola (,), del punto e virgola (;) e della funzione TAB, come specificato nel paragrafo "Controllo della posizione dei caratteri nel buffer di display".

L'istruzione DISP senza operandi cancella il contenuto del buffer di display e posiziona il relativo pointer nella prima posizione. Anche i caratteri visualizzati sul display sono cancellati.

Note

1. Il valore di una espressione numerica è visualizzato preceduto da uno spazio (se è un numero positivo) o dal segno meno (se è un numero negativo) e seguito da uno spazio.

2. I numeri interi sono visualizzati con al massimo 8 cifre significative.

3. I numeri con valore assoluto compreso tra 0.0999999995 e 99999999.4 sono visualizzati, con al massimo 8 cifre significative (se il numero è minore di 1 viene tralasciato lo zero che precede la parte decimale), nel formato in virgola fissa.

4. I numeri con valore assoluto minore di 0.0999999995, rappresentabili con 8 cifre significative, sono visualizzati nel formato in virgola fissa.

5. Tutti gli altri numeri sono visualizzati nel formato in virgola mobile.

6. Il valore di una espressione stringa è visualizzato con la sequenza dei caratteri che la compongono.

Controllo della posizione dei caratteri nel buffer di display

L'esecuzione di una istruzione DISP genera in un registro di 80 byte, detto buffer di display, la stringa di caratteri corrispondente al valore dell'espressione che compare nella istruzione stessa. Un secondo registro, detto pointer, indica la posizione del buffer in cui deve iniziare la stringa di caratteri suddetta. Ogni volta che un carattere è generato nel buffer di display, il pointer avanza di una posizione. Quando il buffer di display è pieno, la generazione di un successivo carattere avviene dalla prima posizione del buffer ed il precedente contenuto è cancellato completamente.

Quando l'istruzione DISP è eseguita, il contenuto del buffer di display è visualizzato a partire dalla prima posizione del buffer. Se il buffer di display con-

tiene più di 32 caratteri, per visualizzare i rimanenti caratteri si deve premere il tasto [→] eventualmente con [SHIFT] o [REPEAT].

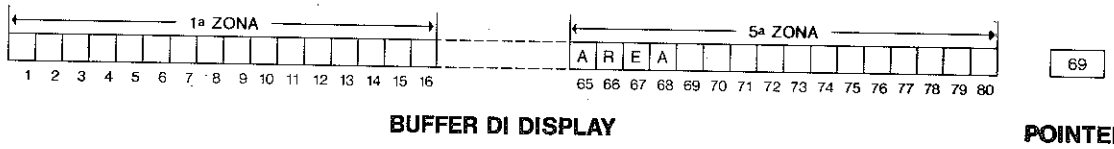


Figura 5-1 Buffer di display e relativo pointer

Ogni volta che una istruzione DISP è eseguita, se la istruzione DISP precedentemente eseguita non ha come carattere finale una virgola (,) od un punto e virgola (;), il contenuto del buffer di display è cancellato e la generazione dei nuovi caratteri riprende dall'inizio.

L'impiego della funzione TAB (num-exp) permette il controllo della posizione dei caratteri nel buffer di display. Il valore dell'espressione numerica num-exp è arrotondato all'intero più prossimo ed assegnato al pointer del buffer di display. Se ad esempio si utilizza in un programma l'istruzione:

```
50 DISP TAB (15); "A"
```

la lettera A sarà visualizzata nella posizione 15 del display che corrisponde alla posizione 15 del buffer di display. Utilizzando le istruzioni in sequenza:

```
50 DISP TAB (50); "A"
60 DELAY 200
```

la lettera A viene generata nella posizione 50 del buffer di display e, premendo i tasti [→] e [SHIFT] contemporaneamente, visualizzata sul display nell'ultima posizione. L'istruzione DELAY mantiene per 20 secondi il carattere A nel buffer di display, permettendone così la visualizzazione sul display.

Come si vede dai suddetti esempi è bene far seguire TAB (num-exp) da punto e virgola (;).

L'impiego della virgola (,) permette di visualizzare i dati distanziati di 16 posizioni l'uno dall'altro. Il buffer di display è diviso in 5 zone, ognuna di 16 posizioni (vedi figura 5-1). In un programma con le seguenti istruzioni:

```
50 DISP
60 DISP"A"
70 DELAY 100
```

l'istruzione 50 pone il pointer del buffer di display in posizione 17 e la successiva istruzione visualizza il carattere A nella posizione 17 del display. In un programma con le seguenti istruzioni:

```
50 DISP "A", "B", "C", "D"
60 DELAY 100
70 DISP "D",
80 DISP "C"
```

l'istruzione 50 genera nel buffer di display il carattere A nella prima posizione, il carattere B nella 17-esima posizione, il carattere C nella 33-esima posizione ed il carattere D nella 49-esima posizione. Sul display A e B appaiono nelle posizioni suddette. Per visualizzare C e D si devono premere i tasti [→] e [REPEAT] contemporaneamente: i due caratteri sono visualizzati distanziati di 16 posizioni sul display. L'istruzione 70 genera il carattere D nella prima posizione del buffer di display e cancella tutti i caratteri precedentemente contenuti nel buffer. Il pointer indirizza la 17-esima posizione del buffer di display (perchè l'istruzione è chiusa con la virgola) e la successiva istruzione DISP visualizza il carattere C in tale posizione.

L'impiego del punto e virgola (;) permette di visualizzare caratteri corrispondenti al valore dell'espressione successiva dalla posizione indirizzata dal pointer in quel momento. Se in un programma si hanno:

```
20 DISP "A"; "B"
30 DELAY 100
40 DISP "C" ;
50 DISP "D"
60 DELAY 100
```

L'istruzione 20 visualizza i caratteri A e B uno di seguito all'altro (nella prima e seconda posizione). Le istruzioni 40 e 50 producono lo stesso risultato per C e D. L'impiego del punto e virgola è particolarmente utile quando l'istruzione DISP è seguita da una istruzione INPUT. In questo caso l'istruzione INPUT mantiene sul display il messaggio precedente seguito da punto interrogativo (?) (vedi nel seguito l'istruzione INPUT).

Nel seguito diamo una tabella che riassume l'impiego della virgola e del punto e virgola come elementi separatori in una istruzione DISP. Il separatore menzionato segue, nella istruzione, il dato specificato nella prima colonna. Nella quarta colonna è indicata la posizione indirizzata dal pointer dopo che i caratteri sono stati inseriti nel buffer.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di display	Posizione indirizzata dal pointer
Espressione numerica	"," virgola	Il valore dell'espressione è registrato nel buffer a partire dalla posizione indirizzata dal pointer, se le posizioni libere sono sufficienti a contenerlo, altrimenti il precedente contenuto del buffer è cancellato ed il valore della espressione è registrato a partire dalla prima posizione.	Il pointer indirizza l'inizio della successiva zona di display.
	";" punto e virgola		Il pointer indirizza la posizione successiva all'ultima occupata dal valore generato nel buffer.
Espressione stringa	"," virgola	La stringa di caratteri corrispondente al valore della espressione è registrata	Il pointer indirizza l'inizio della successiva zona di display.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di display	Posizione indirizzata dal pointer
		nel buffer a partire dalla posizione indirizzata dal pointer, se le posizioni libere sono sufficienti a contenerla, altrimenti il precedente contenuto del buffer è cancellato ed il valore della espressione è registrato a partire dalla prima posizione.	
	";" punto e virgola		Il pointer indirizza la posizione successiva all'ultima occupata dalla stringa generata nel buffer.
stringa nulla	","	Non viene generato alcun carattere nel buffer di display.	Il pointer indirizza l'inizio della successiva zona di display.
	";" punto e virgola		Il pointer non è modificato.

Tabella 5-1 Impiego della virgola e del punto e virgola con DISP

Esempi

1. Nell'esempio seguente si può osservare come vengono visualizzati i valori numerici sul display (che sono anche stampati perchè PRINT ALL è attivo). Il valore della variabile A viene visualizzato arrotondato all'intero più prossimo, perchè il numero delle cifre significative è maggiore di 8, ma in virgola fissa, perchè è compreso nel campo definito nelle note precedenti. I numeri, come si vede, occupano al massimo 14 posizioni del display. Se si utilizza la virgola coma separatore i numeri sono generati nel buffer di display iniziando dalle posizioni: 1,17,33,49 e 65 (per vedere gli ultimi tre si devono usare i tasti SHIFT e →). Se il separatore è il punto e virgola i numeri sono visualizzati interponendo uno spazio tra un valore e l'altro.

```
LIST
FILE

0010 LET A=0.0999999995
0020 LET B=99999999.4
0030 LET C=0.0000999
0040 LET D=9999999999999
0050 LET E=0.000999999999
0060 PRINT "12345678901234567890123456789012345678901234567"
0070 DISP A,-A,B,-B,C
0080 DELAY 200
0090 DISP A;-A;B;-B;C;-C
0100 DELAY 200
0110 PRINT "1234567890"
0120 DISP TAB(9);-D
0130 DELAY 100
0140 PRINT "12345678901234567"
0150 DISP TAB(6),-D
0160 DELAY 100
0170 PRINT "12345678901234567"
0180 DISP ,-E
0190 DELAY 100
0200 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
12345678901234567890123456789012345678901234567
 .10000000       -.10000000        99999999.       -99999999.        .0000999
 .10000000 -.10000000  99999999. -99999999.  .0000999 -.0000999
1234567890
         -1.0000000E+13
12345678901234567
                 -1.0000000E+13
12345678901234567
                 -1.0000000E-03
```

2. Nell'esempio seguente si può vedere che, anche quando i valori da visualizzare non sono dei numeri, l'effetto prodotto dalla virgola è uguale a quello dell'esempio precedente. Se i valori da visualizzare sono separati dal punto e virgola, vedi l'istruzione 120, i caratteri sono visualizzati uno di seguito all'altro. La stringa BRASILE è stata stampata perchè il tasto PRINT ALL è attivo, l'utente non la vede sul display, poichè l'istruzione successiva la cancella.

```
LIST
FILE


0010 PRINT "1234567890123456789012345678901234567890123456789012345678901234567"
0020 DISP "A"
0030 DELAY 50
0040 DISP ,"B"
0050 DELAY 5
0060 DISP ,,"C"
0070 DELAY 100
0080 DISP "INTRODUCI 1";
0090 INPUT A
0100 DISP "INTRODUCI 2",
0110 INPUT B
0120 DISP "A";"B";"C"
0130 DELAY 50
0140 DISP "ZA";
0150 DISP "RA"
0160 DELAY 10
0170 DISP "BRASILE"
0180 DISP
0190 DELAY 50
0200 DISP TAB(9),"AAA"
0210 DELAY 50
0220 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
1234567890123456789012345678901234567890123456789012345678901234567
A
                B
                                C
INTRODUCI 1?
1
INTRODUCI 2     ?
2
ABC
ZARA
BRASILE

                AAA
```

3. Vediamo una routine che impiega l'istruzione DISP per qualificare i dati richiesti dall'istruzione INPUT.

```
LIST
FILE


0010 LET A$="AREA"
0020 LET B$=", VOLUME"
0030 LET C$="PESO"
0040 DISP A$+B$;
0050 INPUT D,E
0060 DISP A$;TAB(10);D;TAB(28);"mq"
0070 DELAY 50
0080 DISP B$,
0090 DISP E;TAB(29);"mc"
0100 DELAY 50
0110 DISP "INTRODUCI PESO SPECIFICO";
0120 INPUT P
0130 DISP C$;TAB(10);P*E;TAB(29);"Kg"
0140 DELAY 50
0150 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
AREA, VOLUME?
10,5
AREA      10                mq
, VOLUME          5          mc
INTRODUCI PESO SPECIFICO?
4
PESO      20                Kg
```

Istruzione DISP USING

Funzione

Visualizza dati e testi, sul display, nel formato specificato dall'utente.

Formato

DISP USING {**line-num** | **string-var**} , {**num-exp** | **string-exp**} [, {**num-exp** | **string-exp**}] ...

dove:

line-num
indica il numero di linea di una istruzione immagine

string-var
indica una variabile stringa il cui contenuto è la immagine con cui i valori specificati nell'istruzione devono essere visualizzati sul display

num-exp
è una espressione numerica il cui valore deve essere visualizzato sul display

string-exp
è una espressione stringa il cui valore deve essere visualizzato sul display

Azione

I valori delle espressioni sono visualizzati, da sinistra a destra, nell'ordine con cui sono specificati nella istruzione, e nelle posizioni specificate dalla immagine di formato definita con l'istruzione il cui numero di linea è line-num o contenuta nella variabile stringa string-var.

Si veda la descrizione dell'istruzione IMMAGINE, nel seguito, per una completa comprensione del modo con cui i valori suddetti sono visualizzati.

1. Se vi sono più espressioni nella istruzione DISP USING che campi di formato nella immagine di formato, i valori in eccedenza sono visualizzati con la stessa immagine di formato iniziando dal primo campo dell'immagine.

2 Se vi sono più campi di formato nella immagine di formato che espressioni nella istruzione DISP USING, in corrispondenza dei campi immagine eccedenti, non sono visualizzati caratteri sul display.

3. Le espressioni presenti nella DISP USING devono essere coerenti con i campi di formato ad essi associati: ad una espressione stringa deve corrispondere un campo immagine di stringa.

Esempi

1; Nel seguente programma si vede come sono visualizzati sul display alcuni valori numerici e stringa impiegando l'istruzione DISP USING.

```
LIST
FILE    *DISPUS

0010 PRINT "Le posizioni del display sono:"
0020 PRINT "123456789012345678901234567890 12"
0030 :########     ####.######
0040 : ##.###↑↑↑↑       $$$####.###
0050 :'LLLLLLLLLLLLLLLLLL    'RRRRRRRR
0060 DISP " UN INTERO ED UN DECIMALE";
0070 GOSUB 220
0080 INPUT A,B
0090 GOSUB 220
0100 DISP USING 30,A,A*B
0110 DELAY 50
0120 GOSUB 220
0130 DISP "NUMERO NEL FORMATO ESPONENZIALE";
0140 INPUT C
0150 GOSUB 220
0160 DISP USING 40,C,A+B
0170 DELAY 50
0180 GOSUB 220
0190 DISP USING 50,"OLIVETTI","P6066"
0200 DELAY 100
0210 GOTO 240
0220 PRINT "Sul display si vede:"
0230 RETURN
0240 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Le posizioni del display sono:
123456789012345678901234567890 12
Sul display si vede:
 UN INTERO ED UN DECIMALE?
15,25.45
Sul display si vede:
     15      381.750000
Sul display si vede:
NUMERO NEL FORMATO ESPONENZIALE?
12E87
Sul display si vede:
  1.200E+88          $  40.450
Sul display si vede:
OLIVETTI                    P6066
```

2. Nel seguente programma si vede che se il numero d campi della immagine di formato è maggiore del numero di valori da visualizzare, i campi immagine eccedenti sono ignorati.

   Se il numero dei valori da visualizzare (istruzione 90 e 100) è maggiore del numero di campi dell'immagine di formato, i valori eccedenti sono visualizzati con la stessa immagine di formato ed i valori precedenti sono cancellati (in questo caso si vedono solo Y &&&&& Z e 1° dato 15 2° dato 16, mentre gli altri rimangono per un tempo brevissimo sul display).

```
LIST
FILE    *DISPU1

0010 : 1o dato  ###    2o dato    ###
0020 : 'LLLLLL        &&&&&     'RRRRRR
0030 PRINT "Le posizioni sul display sono:"
0040 PRINT "123456789012345678901234567890 12"
0050 GOSUB 210
0060 DISP USING 20,"C","D"
0070 DELAY 50
0080 GOSUB 210
0090 DISP USING 20,"W","X","Y","Z"
0100 DELAY 50
0110 : 'L     'L     'L
0120 GOSUB 210
0130 DISP USING 110,"AB"
0140 DELAY 50
0150 GOSUB 210
0160 DISP USING 10,12
0170 DELAY 50
0180 GOSUB 210
0190 DISP USING 10,13,14,15,16
0200 GOTO 230
0210 PRINT "Sul display si vede:"
0220 RETURN
0230 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
Le posizioni sul display sono:
123456789012345678901234567890 12
Sul display si vede:
 C                &&&&&            D
Sul display si vede:
 W                &&&&&            X
 Y                &&&&&            Z
Sul display si vede:
 AB
Sul display si vede:
 1o dato    12     2o dato
Sul display si vede:
 1o dato    13     2o dato      14
 1o dato    15     2o dato      16
```

Istruzione END

Funzione

Definisce la fine di un programma.

Formato

END

Azione

L'esecuzione del programma termina.

Il contenuto delle variabili del programma è cancellato.

I file esterni riferiti nel programma sono chiusi.

Le operazioni di input/output eventualmente in corso di esecuzione sono portate a termine: il contenuto dei buffer relativi ai file dati esterni è registrato nel file esterno; il contenuto dei buffer di stampa è stampato.

Note

1. In un programma l'istruzione END deve essere sempre presente.

2. Dopo l'esecuzione della istruzione END il programma rimane in memoria principale e sul display appare il messaggio "READY", se non è stato inibito quando è stato registrato nella libreria con il comando SAVE (vedi capitolo 3).

3. Se dopo l'esecuzione dell'istruzione END, si introduce il comando FETCH senza operandi, è trasferita nel buffer di tastiera, e visualizzata, la prima istruzione del programma.

4. L'istruzione END deve avere il numero di linea più alto.

5. In un programma deve esservi una sola istruzione END.

Istruzione FILE:

Funzione

Chiude l'accesso di un programma ad un file esterno e lo apre ad un file specificato.

Formato

FILE:file-designator,{string-exp / *}

dove:

file-designator
indica una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica un designatore di file

string-exp
è una espressione stringa il cui valore indica il nome di un file esterno presente in una libreria

*
chiude l'accesso del programma al file a cui era stato assegnato il numero designatore di file specificato con file-designator.

Azione

Al file esterno il cui nome corrisponde al valore della espressione stringa specificata con string-exp è associato il designatore di file indicato con il valore, arrotondato all'intero più prossimo, dell'espressione numerica specificata con file-designator. In questo modo le istruzioni di programma riferentesi al designatore di file suddetto, che sono eseguite successivamente, agiranno sul file con il nome specificato dal valore di string-exp. Il file che era prima associato al designatore di file suddetto è chiuso all'accesso da parte del programma, per cui le istruzioni che sono eseguite successivamente non possono riferirsi ad esso.

Se il secondo operando della istruzione è asterisco (*), è chiuso il file il cui numero designatore è specificato con file-designator. Le istruzioni ese-

guite successivamente non possono più far riferimento a detto file. L'area di memoria utilizzata come buffer del file è a disposizione dell'utente.

Note

1. Il file specificato con il valore di string-exp è cercato dal sistema nelle librerie aperte, secondo l'ordine di apertura. Non deve essere un file già aperto all'accesso da parte del programma.

2. Il file con il nome specificato nella istruzione deve essere stato creato con un comando CREATE (vedi capitolo 3).

3. Utilizzando l'istruzione FILE: un programma può accedere successivamente ad un numero indefinito di file dati esterni.

4. E' talvolta utile poter chiudere l'accesso di un programma ad un file esterno utilizzando l'opzione asterisco (*). Questa operazione protegge il file esterno nel caso di malfunzionamento del sistema (es. caduta di tensione).

5. Il valore della espressione numerica, file-designator, arrotondato all'intero più prossimo, deve essere maggiore di zero e non superiore al numero massimo di file che possono essere elaborati contemporaneamente dal programma (specificato con l'istruzione FILES).

Esempio

Nella routine sottostante si mostra come l'impiego dell'istruzione FILE: permette di assegnare ai designatori di file dei nuovi nomi di file. Le prime istruzioni WRITE: e READ: si riferiscono ai file A ed A1 che sono stati aperti dall'istruzione FILES.

```
LIST
FILE    +FILES1

0010 SCRATCH :1
0020 WRITE :1,"Ho scritto nel file A."
0030 WRITE :1,"Ora ho letto queste due frasi dal file A."
0040 SCRATCH :2
0050 WRITE :2,"Ho scritto nel file A1."
0060 WRITE :2,"Ora ho letto queste ultime due frasi dal file A1."
0070 RESTORE :1
0080 RESTORE :2
0090 DCL 80(A$,B$)
0100 READ :1,A$,B$
0110 PRINT A$
0120 PRINT B$
0130 READ :2,A$,B$
0140 PRINT
0150 PRINT A$
0160 PRINT B$
0170 FILE : 1,"A2"
0180 SCRATCH :1
0190 WRITE :1,"Ho scritto nel file A2."
0200 WRITE :1,"Ora ho letto queste ultime due frasi dal file A2."
0210 RESTORE :1
0220 READ :1,A$,B$
0230 PRINT
0240 PRINT A$
0250 PRINT B$
0260 FILE : 2,"A3"
0270 FILES A;A1
0280 SCRATCH :2
0290 WRITE :2,"Ho scritto nel file A3."
0300 WRITE :2,"Ora ho letto queste ultime due frasi dal file A3."
0310 RESTORE :2
0320 READ :2,A$,B$
0330 PRINT
0340 PRINT A$
0350 PRINT B$
0360 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ho scritto nel file A.
Ora ho letto queste due frasi dal file A.

Ho scritto nel file A1.
Ora ho letto queste ultime due frasi dal file A1.

Ho scritto nel file A2.
Ora ho letto queste ultime due frasi dal file A2.

Ho scritto nel file A3.
Ora ho letto queste ultime due frasi dal file A3.
```

Istruzione FILES

Funzione

Specifica quanti file esterni possono essere elaborati contemporaneamente dal programma.

Formato

FILES { **filename** / * } [,{ **filename** / * }] ...

dove:

filename

indica il nome di un file esterno che deve essere elaborato dal programma

*

permette di aprire un file esterno specificato in una istruzione FILE:

Azione

I file esterni, specificati nella istruzione con il nome filename, sono resi accessibili al programma per operazioni di:

- lettura (file sequenziali)
- lettura e/o registrazione (file ad accesso diretto)

Ad ogni file viene associato un numero intero positivo, detto designatore di file, che coincide con il numero d'ordine con cui il nome del file compare nell'ambito della istruzione FILES (l'ordine si intende da sinistra a destra): il primo nome di file avrà come numero designatore 1, il secondo 2, etc....

Se nella istruzione compaiono uno o più asterischi anche ad essi è assegnato un numero designatore di file, corrispondente al numero d'ordine con cui compaiono nell'istruzione. A tale designatore di file può essere associato un file mediante una istruzione FILE: (vedi l'istruzione FILE:).

Note

1. L'istruzione FILES non è una istruzione di tipo eseguibile: dichiara quanti file dati esterni possono essere elaborati dal programma e con quali numeri designatori essi saranno riferiti nelle istruzioni di programma.

2. In un programma deve esservi una sola istruzione FILES.

3. In una istruzione FILES lo stesso nome di file deve comparire una sola volta.

4. I file con i nomi specificati nella istruzione devono essere stati creati con un comando CREATE (vedi capitolo 3).

5. Il numero di file che possono essere elaborati contemporaneamente dal programma è dato dal numero di operandi specificato nell'istruzione FILES.

6. Il sistema ricerca i file specificati con l'istruzione nelle librerie aperte secondo l'ordine di apertura.

Esempi

1. La routine sottostante mostra che l'istruzione FILES può essere posta in qualunque punto di un programma.

```
LIST
FILE

0010 SCRATCH :1
0020 WRITE :1,"Ho scritto nel file A."
0030 WRITE :1,"Ora ho letto queste due frasi dal file A."
0040 SCRATCH :2
0050 WRITE :2,"Ho scritto nel file A1."
0060 WRITE :2,"Ora ho letto queste ultime due frasi dal file A1."
0070 RESTORE :1
0080 RESTORE :2
0090 DCL 80(A$,B$)
0100 READ :1,A$,B$
0110 PRINT A$
0120 PRINT B$
0130 READ :2,A$,B$
0140 PRINT
0150 PRINT A$
0160 PRINT B$
0170 FILES A;A1
0180 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
Ho scritto nel file A.
Ora ho letto queste due frasi dal file A.

Ho scritto nel file A1.
Ora ho letto queste ultime due frasi dal file A1.
```

2. Nella routine sottostante si vede come si possa ottenere lo stesso risultato della routine precedente utilizzando una istruzione FILES con asterischi e due istruzione FILE:.

```
LIST
FILE    +FILES1

0010 FILE : 1,"A"
0020 SCRATCH :1
0030 WRITE :1,"Ho scritto nel file A."
0040 WRITE :1,"Ora ho letto queste due frasi dal file A."
0050 FILE : 2,"A1"
0060 SCRATCH :2
0070 WRITE :2,"Ho scritto nel file A1."
0080 WRITE :2,"Ora ho letto queste ultime due frasi dal file A1."
0090 RESTORE :1
0100 RESTORE :2
0110 DCL 80(A$,B$)
0120 READ :1,A$,B$
0130 PRINT A$
0140 PRINT B$
0150 READ :2,A$,B$
0160 PRINT
0170 PRINT A$
0180 PRINT B$
0190 FILES *;*
0200 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ho scritto nel file A.
Ora ho letto queste due frasi dal file A.

Ho scritto nel file A1.
Ora ho letto queste ultime due frasi dal file A1.
```

Istruzione FKEY#

Funzione

Assegna un contenuto ai tasti funzione.

Formato

FKEY# n, string-constant

dove:

n

è un numero intero compreso tra 1 e 16

string-constant

è una stringa di caratteri del set ISO.

Azione

Al tasto funzione specificato con n viene assegnata la stringa di caratteri specificata con string-constant. Quando, durante l'esecuzione del programma, viene premuto il tasto indicato con n, la stringa string-constant è introdotta nel buffer di tastiera e visualizzata sul display.

Note

1. Per introdurre nel buffer di tastiera le stringhe associate con le funzioni da F9 ad F16 si devono premere i relativi tasti insieme con il tasto [SHIFT]. Per poter premere i tasti funzione l'esecuzione del programma deve essere interrotta mediante l'istruzione STOP (vedi l'esempio seguente).

2. Il carattere due punti (:) usato come ultimo carattere di string-constant è interpretato come comando di END OF LINE.

3. Ai tasti funzione si può assegnare un comando START line-num che è eseguito quando si preme il corrispondente tasto, se l'ultimo carattere assegnato è due punti: ad esempio START 100: (vedi anche l'esempio seguente).

3. Ai tasti funzione si può assegnare un comando START line-num che è eseguito quando si preme il corrispondente tasto, se l'ultimo carattere assegnato è due punti: ad esempio START 100: (vedi anche l'esempio seguente).

4. I tasti funzione mantengono il contenuto ad essi assegnato finchè non viene effettuata una nuova assegnazione da un altro programma o durante lo stato di calcoli immediati, oppure il sistema è spento.

5. Quando viene acceso il sistema o sono eseguiti i comandi CONFIGURE, LDKEYS, OPTIONS (vedi capitolo 3), ai tasti funzione è associato il contenuto registrato sul disco sistema o floppy disk sistema mediante il comando STKEYS (vedi capitolo 3).

6. La somma dei caratteri di tutte le stringhe associate a tutti i tasti funzione non può essere maggiore di 238.

Esempi

1. Il programma seguente mostra un tipico impiego dei tasti funzione. Dopo aver assegnato ai tasti funzione un contenuto l'istruzione STOP interrompe la esecuzione del programma e l'operatore può premere il tasto funzione che preferisce. In questo caso premendo uno dei tasti funzione specificati (F1, F2 o F9) il sistema esegue una delle tre routine che compongono il programma. Si noti come avendo messo il punto e virgola alla fine della istruzione DISP (numero di linea 130), il messaggio relativo rimane sul display dopo l'esecuzione dell'istruzione STOP. Come si vede sono state riportate le esecuzioni delle tre routine del programma; per l'ultima si lascia al lettore la ricerca della risposta esatta al quesito posto dal sistema ...

```
LIST
FILE    +FKEY

0010 FKEY #1,START 500:
0020 FKEY #2,START 600:
0090 FKEY #9,START 820:
0110 DISP "Quale routine vuoi eseguire?"
0120 DELAY 50
0130 DISP "F1,F2,F9?";
0140 STOP
0500 PRINT "Io sono la routine che inizia dal numero di linea 500 e ti stampo "
0510 PRINT "la tabella ISO in righe e colonne,come puoi vedere:"
0520 PRINT
```

```
0530 FOR I=0 TO 15 STEP 1
0540 FOR J=I TO 255 STEP 16
0550 PRINT TAB((INT(J/16)+1)*5);CHR$(J);
0560 NEXT J
0570 NEXT I
0580 PRINT
0590 GOTO 1500
0600 PRINT "Io sono la routine che ti saluta,guarda il display!"
0605 PRINT "Premi il tasto PRINTALL per non rallentare la visualizzazione."
0610 DCL 256 A$
0620 LET A$=""
0630 FOR I=1 TO 15 STEP 1
0640 READ B$
0650 LET A$=A$+B$
0660 NEXT I
0670 LET L=LEN(A$)
0680 FOR J=1 TO 10000 STEP 1
0690 FOR R=1 TO L-31 STEP 1
0700 DISP EXT$(A$,R,R+31)
0710 FOR X=1 TO 30 STEP 1
0720 LET A=5+9
0730 NEXT X
0740 NEXT R
0750 DISP
0760 NEXT J
0770 DATA "****************"
0780 DATA "Salve, come st","ai? Come vedi l'","istruzione FKEY"
0790 DATA " ti permette la ","realizzazione di"," un programma ","con molte "
0800 DATA "routine."," Tu scegli ...","  quella che ","vuoi eseguire."
0802 DATA " Per interromp","ermi premi BREAK","...SALUTE!"
0810 GOTO 1500
0820 PRINT " Io sono la routine che ti chiede un indovinello."
0830 PRINT
0840 PRINT "Un uomo vuole entrare in un castello ma non conosce la parola "
0850PRINT"d'ordine. Decide di nascondersi dietro ad un albero ed aspettare ..."
0860 PRINT " ...  Dopo un ora arriva un brigante e dal castello una voce "
0870 PRINT "dice :-DIECI?- ...  il brigante risponde :-CINQUE!-"
0875 PRINT "... la porta si apre ed il brigante entra."
0880 PRINT " ...  Dopo un ora arriva un  altro brigante. La voce dal castello "
0890 PRINT "dice:-OTTO?-. Il brigante risponde :-QUATTRO!- ... la porta si  "
0900 PRINT "apre ed il brigante entra."
0910 PRINT "Dopo un altra ora ecco arrivare un altro brigante. "
0920 PRINT "La voce dice:-SEI?- ... Il brigante risponde :-TRE!- ... ed entra. "
0930 PRINT "L'uomo appostato dietro l'albero ora e' sicuro di poter entrare nel"
0940 PRINT "castello. Dopo un ora si presenta davanti al castello ... sente "
0950 PRINT "la voce che dice:-QUATTRO?- ...  lui immediatamente risponde:-DUE!-"
0960 PRINT "Dal castello escono due briganti, lo prendono e ... lo impiccano!!!"
0970 PRINT "MA ALLORA COSA DOVEVA RISPONDERE? ... "
0975 PRINT "DIGITA UNA PAROLA ... CON I CARATTERI TUTTI IN MAIUSCOLO."
0980 INPUT A$
0990 IF A$="SETTE" THEN 1400
1000 PRINT "Hai sbagliato!"
1110 DISP "Vuoi provare ancora";
1120 INPUT A$
1130 IF A$="SI" THEN 975
1140 GOTO 1500
1400 PRINT "Bravo e' esatta!!!"
1500 END
```

END OF LISTING

```
RUN
Quale routine vuoi eseguire?
F1,F2,F9
START 500
Io sono la routine che inizia dal numero di linea 500 e ti stampo
la tabella ISO in righe e colonne,come puoi vedere:

 ▀  ▄        0  @  P  `  p  ⇕  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▛  ▘  !     1  A  Q  a  q  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▌  ▝  "     2  B  R  b  r  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▙  ▖  #     3  C  S  c  s  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▚  ▗  $     4  D  T  d  t  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▜  ▞  %     5  E  U  e  u  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▟  ▟  &     6  F  V  f  v  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▗  ▘  '     7  G  W  g  w  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ↖  ▚  (     8  H  X  h  x  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 →  ▞  )     9  I  Y  i  y  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ≡  ▚  *     :  J  Z  j  z  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ↓  ▘  +     ;  K  [  k  {  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▚  ▝  ,     <  L  \  l  |  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ←  ▖  -     =  M  ]  m  }  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▗  ▗  .     >  N  ↑  n  ¯  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒
 ▗  ▖  /     ?  O  _  o  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒  ▒


RUN
Quale routine vuoi eseguire?
F1,F2,F9
START 600
Io sono la routine che ti saluta,guarda il display!
Se PRINTALL e` attivo inibiscilo.





RUN
Quale routine vuoi eseguire?
F1,F2,F9
START 820
 Io sono la routine che ti chiede un indovinello!

Un uomo vuole entrare in un castello ma non conosce la parola
d'ordine. Decide di nascondersi dietro ad un albero ed aspettare ...
 ... Dopo un'ora arriva un brigante e dal castello una voce
dice  : - DIECI? - ... Il brigante risponde: - CINQUE -
... La porta si apre ed il brigante entra.
 ... Dopo un'ora arriva un altro brigante. La voce dal castello
dice: - OTTO? -. Il brigante risponde: - QUATTRO! - ... La porta si
apre ed il brigante entra.
Dopo un'altra ora ecco arrivare un altro brigante.
La voce dice: - SEI? - ... Il brigante risponde: - TRE - ed entra.
L'uomo appostato dietro l'albero ora e` sicuro di poter entrare nel
castello. Dopo un'ora si presenta davanti al castello ... sente
la voce che dice: - QUATTRO? - Lui immediatamente risponde: - DUE! -
Dal castello escono due briganti, lo prendono e ... lo impiccano!!!
MA ALLORA COSA DOVEVA RISPONDERE? ...
DIGITA UNA PAROLA ... CON I CARATTERI TUTTI IN MAIUSCOLO.




?
DUE
Hai sbagliato!
Vuoi provare ancora?
SI
DIGITA UNA PAROLA ... CON I CARATTERI TUTTI IN MAIUSCOLO.
```

```
?
TRE
Hai sbagliato!
Vuoi provare ancora?
NO
```

2. Vediamo un esempio di una routine che permette di elaborare 10 dati "corretti" anche nel caso in cui s'introduca un dato"scorretto". Se l'utente si accorge che un dato introdotto era "scorretto", preme (F1) e quindi introduce il dato "corretto".

```
FILE

0010 LET C=1
0020 FKEY #1,9999999999999:
0030 INPUT A
0040 IF A=9999999999999 THEN 100
0050 LET B=10000*A
0060 PRINT "A=";A,"B=";B
0070 LET C=C+1
0080 IF C>10 THEN 120
0090 GOTO 30
0100 LET C=C-1
0110 GOTO 30
0120 END

END OF LISTING


A= 1            B= 10000
A= 2            B= 20000
A= 3            B= 30000
A= 4            B= 40000
A= 5            B= 50000
A= 45           B= 450000
A= 6            B= 60000
A= 7            B= 70000
A= 8            B= 80000
A= 9            B= 90000
A= 10           B= 100000
```

Istruzione FNEND

Funzione	Termina la definizione di una funzione multilinea.
Formato	**FNEND**
Azione	Vedi DEF/FNEND
Nota	Ogni definizione di funzione multilinea deve essere chiusa con una istruzione FNEND.

Istruzione FOR

Funzione

Inizia l'esecuzione di un ciclo iterativo.

Formato

FOR control-var = num-exp$_1$ TO num-exp$_2$ [STEP num-exp$_3$]

•
•
•

ISTRUZIONI BASIC

•
•
•

NEXT control-var

dove:

control-var
indica una variabile semplice numerica, detta variabile di controllo, che permette di controllare il numero di esecuzioni successive dell'intero insieme di istruzioni comprese tra l'istruzione FOR e l'istruzione NEXT

num-exp$_1$
indica una espressione numerica il cui valore viene assegnato alla variabile di controllo

num-exp$_2$
indica una espressione numerica il cui valore è confrontato con il valore della variabile di controllo per decidere se l'insieme di istruzioni suddette deve essere eseguito o saltato

num-exp$_3$
indica una espressione numerica il cui valore viene aggiunto al valore della variabile di controllo dopo ogni esecuzione dell'insieme delle istruzioni suddette

Azione

Quando è eseguita l'istruzione FOR, alla variabile di controllo è assegnato il valore della espressione numerica specificata con num-exp$_1$. Il valore della espressione numerica specificata con num-exp$_2$ è confrontato

con il valore della variabile di controllo; se quest'ultimo è minore od uguale al primo, sono eseguite le successive istruzioni fino alla istruzione NEXT. Quando è eseguita l'istruzione NEXT, il valore della variabile di controllo è incrementato del valore specificato con l'espressione num-exp$_3$ (detto incremento). La variabile di controllo è quindi confrontata con il valore di num-exp$_2$, se il suo valore è ancora minore od uguale a quest'ultimo sono nuovamente eseguite le istruzioni fino alla NEXT. Il ciclo di operazioni descritto viene ripetuto finchè il valore della variabile di controllo non supera quello di num-exp$_2$. Quando ciò accade, l'esecuzione del programma passa alla prima istruzione eseguibile successiva all'istruzione NEXT. La variabile di controllo mantiene l'ultimo valore assunto.

Se il valore della espressione num-exp$_1$ è negativo, il ciclo FOR/NEXT è eseguito finchè il valore della variabile di controllo non è minore del valore della espressione num-exp$_2$.

Se il valore della espressione num-exp$_3$ è zero allora il ciclo FOR/NEXT viene ripetuto senza fine.

Note

1. Se non si specifica l'opzione STEP num-exp$_3$, il valore da aggiungere come incremento al valore della variabile di controllo è +1.

2. La variabile di controllo control-var specificata nella istruzione NEXT deve essere la stessa che è specificata nell'istruzione FOR.

3. Due o più cicli FOR/NEXT possono essere nidificati ma non intersecati, per cui la seguente routine è corretta:

```
 50 FOR I = 1 TO 10
 - - - - -
100 FOR J = 2 TO 20
- - - - -
200 NEXT J
 - - - - -
 300 NEXT I
```

mentre la seguente routine non è corretta:

```
{ 50 FOR I = 1 TO 10
  - - - - -
  { 100 FOR J = 2 TO 20
  - - - - -
  { 150 NEXT I
  { 200 NEXT J
```

4. Due o più cicli FOR/NEXT nidificati non devono avere la stessa variabile di controllo.

5. In un programma ad una istruzione FOR deve sempre corrispondere una istruzione NEXT.

6. Le istruzioni del programma esterne ad un ciclo FOR/NEXT non possono rinviare l'esecuzione ad istruzioni interne al ciclo suddetto; fa eccezione l'istruzione RETURN come sottospecificato.

7. In un ciclo FOR/NEXT vi possono essere istruzioni di tipo GOSUB ed ON...GOSUB che rinviano l'esecuzione ad istruzioni esterne al ciclo FOR/NEXT. Le relative istruzioni RETURN riportano poi l'esecuzione all'interno del ciclo FOR/NEXT.

8. In un ciclo FOR/NEXT vi possono essere delle istruzioni GOTO, ON...GOTO, IF...THEN che rinviano l'esecuzione ad istruzioni esterne al ciclo FOR/NEXT. In questo caso la variabile di controllo mantiene l'ultimo valore assunto.

9. Si può modificare il valore della variabile di controllo con delle istruzioni di assegnazione (LET) nel ciclo FOR/NEXT.

10. Un ciclo FOR/NEXT non può intersecarsi con una definizione di funzione multilinea, ad esempio non si può avere:

```
50 FOR I = 1 TO 10
60 A = B*C
70 DEF FNA (X,Y)
80 PRINT Z
90 FN* = X*Y
100 NEXT I
120 FNEND
```

11. Se la variabile di controllo, il valore finale e l'incremento sono stati dichiarati in singola precisione, l'esecuzione del ciclo FOR/NEXT è più rapida che nel caso in cui siano rappresentati in doppia precisione.

Esempi

1. La routine seguente mostra l'impiego dell'istruzione FOR senza specificare esplicitamente l'incremento della variabile di controllo. Come si vede il sistema assume un incremento (STEP) implicito pari ad uno. Si osservi che, in questo caso, la variabile di controllo assume l'ultimo valore aumentato di uno quando termina l'esecuzione del ciclo FOR/NEXT relativo.

```
0010 FOR J=1 TO 10
0020 PRINT J,
0030 NEXT J
0040 PRINT
0050 PRINT "J=";J
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1              2              3              4              5
 6              7              8              9              10

J= 11
```

2. Nella routine sottostante, alla variabile di controllo viene assegnato un valore iniziale negativo.

```
LIST
FILE


0010 FOR I=-90 TO 10 STEP 10
0020 PRINT I,
0030 NEXT I
0040 PRINT
0045 PRINT "I=";I
0050 END

END OF LISTING


RUN
-90            -80            -70            -60            -50
-40            -30            -20            -10            0
 10
I= 20
```

3. Nell'istruzione FOR si è specificato un valore finale per la variabile di controllo minore di quello iniziale ma l'incremento è positivo. Il ciclo FOR/NEXT non è eseguito (infatti la variabile di controllo mantiene il valore iniziale e l'istruzione PRINT interna al ciclo non è eseguita).

```
LIST
FILE

0010 FOR I=1 TO -10 STEP 1
0020 PRINT I,
0030 NEXT I
0040 PRINT
0050 PRINT "I=";I
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

I= 1
```

4. Questa volta il valore finale è minore di quello iniziale, ma l'incremento specificato è negativo. Il ciclo FOR/NEXT è eseguito. Si osservi come la variabile di controllo assume il valore finale decrementato di uno (passo), al termine della esecuzione del ciclo FOR/NEXT.

```
LIST
FILE

0010 FOR I=1 TO -10 STEP -1
0020 PRINT I,
0030 NEXT I
0040 PRINT
0050 PRINT "I=";I
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1              0              -1             -2             -3
-4             -5              -6             -7             -8
-9             -10
I=-11
```

5. Nella routine sottostante si vede che i cicli FOR/NEXT possono essere nidificati.

```
LIST
FILE    +ISOTAB

0010 PRINT
0020 PRINT "Ecco i caratteri della tabella ISO ordinati in righe e colonne:"
0030 PRINT
0040 FOR I=0 TO 15 STEP 1
0050 FOR J=I TO 255 STEP 16
0060 PRINT TAB((INT(J/16)+1)*5);CHR$(J);
0070 NEXT J
0080 NEXT I
0090 PRINT
0100 PRINT "I=";I,"J=";J
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

Ecco i caratteri della tabella ISO ordinati in righe e colonne:

    ▒   ▒       0   @   P   `   p   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ┌   ▒   !   1   A   Q   a   q   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ┴   ▒   "   2   B   R   b   r   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ┘   ▒   #   3   C   S   c   s   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ┤   ▒   $   4   D   T   d   t   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ▒   ▒   %   5   E   U   e   u   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ▒   ▒   &   6   F   V   f   v   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ┬   ┤   '   7   G   W   g   w   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ├   ▒   (   8   H   X   h   x   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    │   ┼   )   9   I   Y   i   y   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ≡   ▒   *   :   J   Z   j   z   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ┼   ▒   +   ;   K   [   k   {   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ▒   ▒   ,   <   L   \   l   |   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ▒   ▒   -   =   M   ]   m   }   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ▒   ▒   .   >   N   ↑   n   ~   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
    ▒   ▒   /   ?   O   _   o   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒   ▒
I= 16           J= 271
```

6. In questo ciclo FOR/NEXT si impiega una istruzione di salto (IF), per cui il controllo della esecuzione viene ceduto al progamma principale, quando la condizione specificata si avvera. Si noti come la variabile di controllo assume l'ultimo valore assegnatole prima dell'esecuzione dell'istruzione suddetta.

```
LIST
FILE

0010 FOR I=1 TO 10 STEP 1
0020 PRINT I,
0030 IF I=5 THEN 50
0040 NEXT I
0050 PRINT "I=";I
```

```
0060 PRINT "E' stata esequita l'istruzione IF."
0070 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1                2                3                4
I= 5
E' stata esequita l'istruzione IF.
```

7. Nel ciclo FOR/NEXT seguente si utilizza una istruzione di salto ad un sottoprogramma esterno al ciclo suddetto. Si noti come la variabile di controllo mantiene il valore assunto prima del salto al sottoprogramma.

```
LIST
FILE


0010 FOR I=1 TO 10 STEP 1
0020 GOSUB 80
0030 PRINT I*10
0040 NEXT I
0050 PRINT
0060 PRINT "I=";I
0070 GOTO 100
0080 PRINT "GOSUB numero";I,
0090 RETURN
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
GOSUB numero 1    10
GOSUB numero 2    20
GOSUB numero 3    30
GOSUB numero 4    40
GOSUB numero 5    50
GOSUB numero 6    60
GOSUB numero 7    70
GOSUB numero 8    80
GOSUB numero 9    90
GOSUB numero 10                    100

I= 11
```

8. Nel ciclo FOR/NEXT seguente si vede che il valore della variabile di controllo può essere modificato nel suo interno, mentre il valore finale e l'incremento non possono essere modificati.

```
LIST
FILE


0010 LET F=100
0020 LET P=5
0030 FOR I=1 TO F STEP P
0040 PRINT "I=";I,
0050 LET I=I+5
0060 PRINT "I+5=";I
0070 LET F=G=200
0080 PRINT "F=";F,"P=";P
0090 NEXT I
0100 PRINT "I=";I
0110 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
I= 1              I+5= 6
F= 200            P= 5
I= 11             I+5= 16
F= 200            P= 5
I= 21             I+5= 26
F= 200            P= 5
I= 31             I+5= 36
F= 200            P= 5
I= 41             I+5= 46
F= 200            P= 5
I= 51             I+5= 56
F= 200            P= 5
I= 61             I+5= 66
F= 200            P= 5
I= 71             I+5= 76
F= 200            P= 5
I= 81             I+5= 86
F= 200            P= 5
I= 91             I+5= 96
F= 200            P= 5
I= 101
```

Istruzione GO SUB

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma.

Formato

GO SUB line-num

dove:
line-num

che indica un numero di linea di una istruzione del programma.

Azione

L'esecuzione del programma è trasferita all'insieme di istruzioni (detto sottoprogramma) che inizia con l'istruzione il cui numero di linea è specificato con line-num e termina con l'istruzione RETURN (vedi istruzione RETURN). Quando è eseguita l'istruzione RETURN, l'esecuzione del programma passa alla prima istruzione esecutiva successiva alla istruzione GO SUB.

Note

1. Nel sottoprogramma vi può essere più di una istruzione RETURN, ma, come buona norma di programmazione, è consigliabile l'impiego di una sola istruzione RETURN a cui fare riferimento con istruzioni GO TO da diversi punti del sottoprogramma.

2. Le istruzioni RETURN sono collegate alle istruzioni GO SUB in modo dinamico: se sono eseguite, ad esempio, 10 istruzioni GO SUB, la prima istruzione RETURN che sarà successivamente eseguita rinvierà il controllo della esecuzione del programma alla istruzione esecutiva successiva alla decima istruzione GO SUB eseguita; la seconda istruzione RETURN eseguita rinvierà l'esecuzione del programma alla istruzione esecutiva successiva alla nona istruzione

GO SUB eseguita e così via.

3. Il numero di istruzioni GO SUB che possono essere eseguite prima che sia eseguita una istruzione RETURN è limitato dalla dimensione della memoria utente disponibile.

4. In un sottoprogramma vi può essere una istruzione che trasferisce l'esecuzione alla istruzione GO SUB che richiama il sottoprogramma stesso.

5. In un ciclo FOR/NEXT si possono usare istruzioni GO SUB che si riferiscono ad una istruzione esterna al ciclo suddetto, ma non vi possono essere istruzioni GO SUB che si riferiscono ad una istruzione interna al ciclo.

6. line-num non può essere il numero di linea di una istruzione interna ad un ciclo FOR/NEXT.

7. Un sottoprogramma può far parte di una funzione multilinea; in questo caso anche l'istruzione GO SUB deve essere una istruzione della funzione multilinea.

8. Ogni volta che una istruzione RETURN è eseguita vi deve essere almeno una istruzione GO SUB per la quale non sia stata eseguita la relativa istruzione RETURN.

9. Da un sottoprogramma si può uscire con una istruzione di salto (esempio GO TO, IF... THEN) prima che venga eseguita l'istruzione RETURN, ma è meglio evitare l'impiego di tale possibilità.

Esempi

1. La routine sottostante contiene un sottoprogramma che permette di calcolare il massimo comun divisore di tre numeri introdotti da tastiera.

```
LIST
FILE    +GOSUB

0010 DISP "Valori per A, B e C";
0020 INPUT A,B,C
0030 LET X=A
0040 LET Y=B
0050 GOSUB 140
0060 LET X=G
0070 LET Y=C
0080 GOSUB 140
```

```
0090 PRINT "A=";A,"B=";B,"C=";C,"MCD=";G
0100 DISP "si=SI,no=NO; CONTINUI";
0110 INPUT A$
0120 IF A$="NO" THEN 220
0130 GOTO 10
0140 LET Q=INT(X/Y)
0150 LET R=X-Q*Y
0160 IF R=0 THEN 200
0170 LET X=Y
0180 LET Y=R
0190 GOTO 140
0200 LET G=Y
0210 RETURN
0220 END

END OF LISTING


RUN
Valori per A, B e C?
10,1110,150
A= 10          B= 1110          C= 150          MCD= 10
si=SI,no=NO; CONTINUI?
SI
Valori per A, B e C?
125,45,75
A= 125         B= 45            C= 75           MCD= 5
si=SI,no=NO; CONTINUI?
NO
```

2. Il programma sottostante è composto da due sotto-programmi nidificati che permettono di calcolare il valore medio ed il massimo comun divisore di tre numeri introdotti da tastiera.

```
LIST
FILE    +GOSUB1


0010 DISP "Valori per A, B e C";
0020 INPUT A,B,C
0030 LET X=A
0040 LET Y=B
0050 GOSUB 131
0060 LET X=G
0070 LET Y=C
0080 GOSUB 140
0090 PRINT "A=";A,"B=";B,"C=";C,"MCD=";G
0100 DISP "si=SI,no=N); CONTINUI";
0110 INPUT A$
0120 IF A$="NO" THEN 220
0130 GOTO 10
0131 GOSUB 211
0140 LET Q=INT(X/Y)
0150 LET R=X-Q*Y
0160 IF R=0 THEN 200
0170 LET X=Y
0180 LET Y=R
0190 GOTO 140
0200 LET G=Y
0210 RETURN
0211 LET D=(A+B+C)/3
0212 PRINT "A=";A,"B=";B,"C=";C,"Valore medio =";D
0213 RETURN
0220 END

END OF LISTING
```

```
RUN
Valori per A, B e C?
10,20,30
A= 10          B= 20          C= 30          Valore medio = 20
A= 10          B= 20          C= 30          MCD= 10
si=SI,no=NO; CONTINUI?
SI
Valori per A, B e C?
120,150,300
A= 120         B= 150         C= 300         Valore medio = 190
A= 120         B= 150         C= 300         MCD= 30
si=SI,no=NO; CONTINUI?
NO
```

3. In questo esempio, dopo l'istruzione GOSUB con numero di linea 10, è eseguita quattro volte l'istruzione GOSUB con numero di linea 70. L'istruzione RETURN rinvia per quattro volte all'esecuzione dell'istruzione 72 ed, infine, all'istruzione 30.

```
LIST
FILE    +GOSUB4

0005 LET A=0
0010 PRINT "Vediamo un esempio di sottoprogramma che richiama se stesso."
0020 GOSUB 50
0030 GOTO 30
0050 LET A=A+1
0055 IF A>5 THEN 80
0060 PRINT "Esecuzione n";A;"del sottoprogramma."
0070 GOSUB 50
0072 LET A=A-1
0075 PRINT "Eseguita la RETURN relativa al GOSUB n";A
0080 RETURN
0090 END

END OF LISTING


RUN
Vediamo un esempio di sottoprogramma che richiama se stesso.
Esecuzione n 1 del sottoprogramma.
Esecuzione n 2 del sottoprogramma.
Esecuzione n 3 del sottoprogramma.
Esecuzione n 4 del sottoprogramma.
Esecuzione n 5 del sottoprogramma.
Eseguita la RETURN relativa al GOSUB n 5
Eseguita la RETURN relativa al GOSUB n 4
Eseguita la RETURN relativa al GOSUB n 3
Eseguita la RETURN relativa al GOSUB n 2
Eseguita la RETURN relativa al GOSUB n 1
```

Istruzione GOTO

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata.

Formato

GOTO line-num

dove:
line-num
è un numero intero positivo compreso tra 1 e 9999 che indica il numero di linea di una istruzione del programma.

Azione

L'esecuzione del programma passa alla istruzione il cui numero di linea è specificato con line-num.

Note

1. Se line-num è il numero di linea di una istruzione non esecutiva allora l'esecuzione del programma passa alla successiva istruzione esecutiva.

2. Una istruzione GO TO esterna ad una definizione di funzione multilinea non può avere come operando, line-num, un numero di linea di una istruzione interna alla definizione di funzione suddetta.

3. Una istruzione GOTO interna ad una definizione di funzione multilinea, non deve avere come operando, line-num, un numero di linea di una istruzione esterna alla definizione di funzione suddetta.

4. Una istruzione GOTO esterna ad un ciclo FOR/NEXT non può avere come operando, line-num, un numero di linea di una istruzione interna al ciclo suddetto.

5. Una istruzione GOTO interna ad un ciclo FOR/NEXT

può avere come operando, line-num, un numero di linea di una istruzione esterna al ciclo suddetto.

6. Come si vede nei testi successivi, la parola chiave può avere o no interposto uno spazio, quindi si può specificare : GO TO o GOTO.

Esempi

1. Vediamo un esempio d'impiego dell'istruzione GOTO.

```
LIST
FILE

0010 PRINT "Quando vuoi fermarmi premi BREAK!"
0020 PRINT
0030 PRINT
0040 PRINT TAB(INT(RND*1000)):CHR$(RND*10+1)
0050 GOTO 40
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Quando vuoi fermarmi premi BREAK!
```

2. Come si vede è possibile dall'interno di un ciclo FOR/NEXT rinviare l'esecuzione all'esterno del ciclo stesso.

```
LIST
FILE


0010 FOR I=1 TO 10 STEP 1
0020 PRINT I
0030 GOTO 50
0040 NEXT I
0050 PRINT "Sono uscito da un ciclo FOR/NEXT!"
0060 END

END OF LISTING


RUN
 1
Sono uscito da un ciclo FOR/NEXT!
```

Istruzione IF...THEN

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata, nel caso che si verifichi la condizione predefinita.

Formato

```
IF { ({num-exp1 rel-opr num-exp2 | string-exp1 rel-opr string-exp2}) {AND | OR} ({num-exp3 rel-opr num-exp4 | string-exp3 rel-opr string-exp4}) | {num-exp1 rel-opr num-exp2 | string-exp1 rel-opr string-exp2} } THEN line-num
```

dove:

$num\text{-}exp_1$

è una espressione numerica che viene confrontata con l'espressione numerica specificata con $\underline{num\text{-}exp}_2$

$string\text{-}exp_1$

è una espressione di tipo stringa che viene confrontata con $\underline{string\text{-}exp}_2$

rel-opr

è un operatore di confronto scelto tra uno dei seguenti:

<>	oppure	><	non uguale
	=		uguale
>=	oppure	=>	maggiore o uguale
<=	oppure	=<	minore o uguale
	>		maggiore di
	<		minore di

$num\text{-}exp_2$

è una espressione numerica che viene confrontata con $num\text{-}exp_1$

$string\text{-}exp_2$

è una espressione di tipo stringa che viene confrontata con $\underline{string\text{-}exp}_1$

AND

specifica l'operazione di <u>congiunzione</u> tra due proposizioni logiche

OR

specifica l'operazione di <u>disgiunzione</u> tra due proposizioni logiche

num-exp$_3$
è una espressione numerica che viene confrontata con l'espressione numerica specificata con num-exp$_4$

string-exp$_3$
è una espressione stringa che viene confrontata con l'espressione stringa specificata con string-exp$_4$

num-exp$_4$
è una espressione numerica che viene confrontata con l'espressione numerica specificata con num-exp$_3$

string-exp$_4$
è una espressione stringa che viene confrontata con l'espressione stringa specificata con string-exp$_3$

line-num
è un numero intero positivo che indica il numero di linea di una istruzione del programma.

Azione

Se è specificato l'operando AND, le quattro espressioni specificate sono calcolate ed i loro valori sono confrontati a due a due come indicato dai rispettivi operatori di confronto. Se entrambe le relazioni sono soddisfatte, l'esecuzione del programma passa all'istruzione con numero di linea specificato con line-num. In ogni altro caso (una sola delle due relazioni od entrambe non sono soddisfatte) l'esecuzione del programma prosegue con l'istruzione esecutiva successiva alle istruzioni IF...THEN.

Se è specificato l'operando OR, le quattro espressioni specificate sono calcolate ed i loro valori sono confrontati a due a due come indicato dai rispettivi operatori di confronto. Se una sola od entrambe le relazioni sono soddisfatte, l'esecuzione del programma passa all'istruzione con numero di linea specificato con line-num. Se nessuna delle due relazioni è soddisfatta, l'esecuzione del programma prosegue con l'istruzione esecutiva successiva alle istruzioni IF... THEN.

Se non è specificato nè l'operando AND nè l'operando OR, le due espressioni sono calcolate ed i valori ottenuti sono confrontati secondo l'operatore di confronto specificato. Se la relazione è soddisfatta l'esecuzione del programma passa all'istruzione con numero di linea specificato con line-num. Se la relazione non è soddisfatta l'esecuzione del programma prosegue con l'istruzione esecutiva successiva alla istruzione IF...THEN.

Note

1. L'impiego dell'operatore di uguaglianza tra due espressioni numeriche in una istruzione IF...THEN può, in alcuni casi, dar luogo a degli inconvenienti a causa della rappresentazione dei numeri che è necessariamente limitata. Ad esempio se in un programma ci sono delle istruzioni come:

```
10   A = 1.0/3 *3
20   IF A = 1 THEN 75
```

allora l'istruzione 20 non trasferirà il controllo all'istruzione 75 perchè il valore assegnato ad A non è 1 ma 0.9999999999999 oppure 0.999999 a seconda del tipo di precisione dichiarata per A (doppia o singola).

2. Una istruzione IF...THEN esterna ad una definizione di funzione multilinea non può avere come operando, line-num, un numero di linea di una istruzione interna alla definizione di funzione suddetta.

3. Una istruzione IF...THEN che fa parte di una definizione di funzione multilinea non può avere come operando un numero di linea, line-num, di una istruzione che è esterna alla definizione di funzione suddetta.

4. Una istruzione IF...THEN esterna ad un ciclo FOR/NEXT non può avere come operando, line-num, un numero di linea di una istruzione interna al ciclo suddetto.

5. Le espressioni confrontate devono essere omogenee: entrambe numeriche od entrambe stringa.

Esempi

1. Vediamo un esempio di impiego dell'istruzione IF...THEN in cui viene fatto un confronto fra stringhe.

```
LIST
FILE    +IFTHEN

0010PRINT"Ecco un esempio di istruzione IF in cui si confrontano delle stringhe"
0020 PRINT
0030 DISP "Introduci il nome di una citta`!  "
0040 INPUT A$
0050 IF A$="MILANO" THEN 80
0060 PRINT "Non e` la citta` di cui voglio parlarti. Digitane un altra."
```

```
0070 GOTO 40
0080 PRINT "Milano e` proprio la citta` di cui voglio parlarti."
0090 PRINT "Ma ora non ho molto tempo per cui ti saluto ... ."
0100 PRINT " ... ne riparleremo un altra volta."
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio di istruzione IF in cui si confrontano delle stringhe

Introduci il nome di una citta`!
?
ROMA
Non e` la citta` di cui voglio parlarti. Digitane un altra.
?
MILANO
Milano e` proprio la citta` di cui voglio parlarti.
Ma ora non ho molto tempo per cui ti saluto ...
 ... ne riparleremo un altra volta.
```

2. Vediamo un esempio in cui nell'istruzione IF...THEN vi è un confronto tra valori numerici.

```
LIST
FILE


0010 DISP "Introduci quanto guadagni!      ";
0020 INPUT A
0030 IF A<=1000000 THEN 70
0040 IF A<=5000000 THEN 90
0050 IF A<=10000000 THEN 110
0060 PRINT "Guadagni troppo per cui pagherai";A*50/100;"lire di tasse!"
0070 PRINT "Non guadagni molto per cui pagherai";A*1/100;"lire di tasse."
0080 GOTO 120
0090 PRINT "Guadagni abbastanza per cui pagherai";A*5/100;"lire di tasse."
0100 GOTO 120
0110 PRINT "Guadagni parecchio per cui pagherai";A*10/100;"lire di tasse."
0120 END

END OF LISTING


RUN
Introduci quanto guadagni!      ?
650000
Non guadagni molto per cui pagherai 6500 lire di tasse.
RUN
Introduci quanto guadagni!      ?
1500000
Guadagni abbastanza per cui pagherai 75000 lire di tasse.
RUN
Introduci quanto guadagni!      ?
5600000
Guadagni parecchio per cui pagherai 560000 lire di tasse.
RUN
Introduci quanto guadagni!      ?
11000000
Guadagni troppo per cui pagherai 5500000 lire di tasse!
Non guadagni molto per cui pagherai 110000 lire di tasse.
```

3. Vediamo come funziona la seguente routine che impiega una istruzione IF...THEN con l'operando AND.

```
LIS
FILE

0010 INPUT A,B
0020 INPUT A$,B$
0030 IF (A=B)AND (A$=B$) THEN 60
0040 PRINT "L'esecuzione e' continuata in sequenza!"
0050 GOTO 70
0060 PRINT "L'esecuzione non e' continuata in sequenza!"
0070 GOTO 10
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
?
10,10
?
PARIGI,PARIGI
L'esecuzione non e' continuata in sequenza!
?
10,10
?
PARIGI,ROMA
L'esecuzione e' continuata in sequenza!
?
10,12
?
PARIGI,PARIGI
L'esecuzione e' continuata in sequenza!
?
10,12
?
PARIGI,ROMA
L'esecuzione e' continuata in sequenza!
?
```

4. Diamo qui sotto un esempio d'impiego della istruzione IF...THEN con l'operando OR.

```
FILE

0010 INPUT A,B
0020 INPUT A$,B$
0030 IF (A=B)OR (A$=B$) THEN 60
0040 PRINT "L'esecuzione e' continuata in sequenza!"
0050 GOTO 70
0060 PRINT "L'esecuzione non e' continuata in sequenza!"
0070 GOTO 10
0080 END

END OF LISTING
```

```
RUN
?
10,10
?
PARIGI,PARIGI
L'esecuzione non e' continuata in sequenza!
?
10,10
?
PARIGI,ROMA
L'esecuzione non e' continuata in sequenza!
?
10,12
?
PARIGI,PARIGI
L'esecuzione non e' continuata in sequenza!
?
10,12
?
PARIGI,ROMA
L'esecuzione e' continuata in sequenza!
?
```

Istruzione IMMAGINE

Funzione

Specifica un formato predefinito utilizzato dalla istruzioni PRINT USING, DISP USING, BUILD USING, MAT PRINT USING.

Formato

: { literal-field | image-field } [literal-field | image-field] ...

dove:

literal-field

è un campo costituito da caratteri che sono riprodotti esattamente come sono specificati

image field

è un campo costituito da caratteri che sono sostituiti con i valori delle espressioni specificate nella istruzione che fa riferimento alla istruzione immagine suddetta.

Azione

Ogni volta che una istruzione PRINT USING, MAT PRINT USING, DISP USING o BUILD USING, contenente come operando il numero di linea della istruzione IMMAGINE di formato, è eseguita, i caratteri da stampare, visualizzare su display o trasferire in una variabile stringa, sono generati rispettivamente nel buffer di stampa, nel buffer di display o nella variabile stringa con il formato descritto nel paragrafo "Formato associato ai valori delle espressioni presenti nelle istruzioni PRINT USING, DISP USING, BUILD USING".

Campi immagine di formato (image field)

I "campi immagine" presenti in una istruzione IMMAGINE di formato possono essere campi "immagine di numero", associati ai valori di espressioni numeriche, o campi "immagine di stringa", associati ai valori di espressioni stringa.

I campi "immagine di numero" possono essere:

campi "immagine di numero intero"
campi "immagine di numero decimale"
campi "immagine di numero esponenziale"
campi "immagine con segno $"

1. Campo "immagine di numero intero" - è costituito da una sequenza di segni di numero

   <u>formato</u>: # #[#]...

   <u>estensione</u>: minimo 2 massimo 25 caratteri

2. Campo "immagine di numero decimale"- è composto da una sequenza di segni di numero # e dal punto decimale (.)

   <u>formato</u>: # # [#]...[#]...

   <u>estensione</u>: minimo 3 massimo 26 caratteri

3. Campo "immagine di numero esponenziale"- è composto da una sequenza di segni di numero #, dal punto decimale e dal simbolo immagine dell'esponente

   <u>formato</u>: campo "immagine di numero decimale" ↑↑↑↑

   <u>estensione</u>: minimo 7 massimo 30 caratteri

   <u>nota</u>: il simbolo immagine dell'esponente è sempre l'ultimo simbolo nel campo numerico esponenziale.

4. Campo "immagine con segno $" - è costituito da due segni $ oppure da uno o più segni di $ seguiti, eventualmente, da un campo "immagine di numero intero", o da un campo "immagine di numero decimale", o da un punto decimale seguito da uno o più segni di numero #.

   formato: $ [$] ... [campo "immagine di numero intero" / campo "immagine di numero decimale" / .#[#]...]

estensione: minimo 2 massimo 30 caratteri

I campi "immagine di stringa" sono costituiti dal segno di apostrofo seguito eventualmente da una o più lettere L o R o C.

formato: $\left\{ \begin{array}{l} '[L] \cdots \\ '[R] \cdots \\ '[C] \cdots \end{array} \right\}$

estensione: minimo 1 massimo 74 caratteri

Formato associato ai valori delle espressioni presenti nelle istruzioni PRINT USING, MAT PRINT, DISP USING e BUILD USING

La conversione dei valori delle espressioni presenti nelle istruzioni PRINT USING, MAT PRINT USING, DISP USING e BUILD USING è realizzata secondo le seguenti regole:

1. Il valore numerico associato ad un campo "immagine di numero intero" è generato con ogni cifra in corrispondenza di ogni segno # . Il numero è allineato a destra all'interno del campo. Se il numero non è intero viene troncata la parte decimale. Se il numero è positivo viene anteposto uno spazio alla prima cifra. Se il numero è negativo viene anteposto un segno meno alla prima cifra. Se il numero ha, nella parte interna, più cifre dei segni # che costituiscono il campo, è generato un asterisco in corrispondenza di ogni segno # del campo.

2. Il valore numerico associato ad un "campo immagine di numero decimale" è generato con ogni cifra in corrispondenza di un segno # . Il punto decimale è nella stessa posizione in cui è indicato nel campo immagine. La parte intera del numero è generata a sinistra del punto decimale. La parte decimale è generata a destra del punto decimale. Se il numero ha più cifre decimali di quanti sono i segni # dopo il punto decimale, è arrotondato e quindi troncato dalle cifre eccedenti il campo. Se il numero è positivo viene anteposto uno spazio alla prima cifra. Se il numero è negativo, un segno meno è anteposto alla prima cifra. Se il numero ha più cifre intere di quanti sono i segni # prima del punto decimale, è generato un asterisco in corrispondenza di ogni carattere del campo.

3. Il valore numerico associato ad un campo "immagine di un numero esponenziale" è generato con le stesse modalità indicate nel punto 2. I segni ↑↑↑↑ sono sostituiti rispettivamente con il carattere E, il segno + o - e due cifre da 0 a 9 che indicano l'esponente in base 10 della potenza per la quale è moltiplicato il numero decimale che precede la sostituzione di ↑↑↑↑.

4. Il valore numerico associato ad un campo con segno $ viene generato anteponendo ad esso un segno $. Il numero è allineato nell'ambito del campo secondo le modalità descritte nei punti 1 e 2. Se il campo immagine è composto da soli segni di $, il valore numerico deve essere intero (se non lo è viene troncato nella parte decimale) e viene generato allineato a destra nell'ambito del campo. Se il campo immagine è composto da più segni di $ seguiti da un campo immagine numerico ed il valore numerico ad esso associato è costituito da un numero di cifre inferiore al campo numerico suddetto, il segno di $ viene generato nell'ultima posizione a destra in cui compare nel campo immagine. Se il campo immagine è composto da più segni di $ seguiti da un campo immagine numerico ed il valore numerico associato è costituito da un numero di cifre superiore, nella parte intera, alla parte del campo numerico a sinistra del punto decimale, i segni di $ diversi dal primo sono considerati come dei segni di #.

5. Il valore stringa associato ad un campo immagine di stringa viene generato allineato a sinistra, a destra od al centro, nell'ambito del campo suddetto, a seconda che dopo l'apostrofo vi siano rispettivamente la lettera L, R o C. Se la stringa eccede il campo immagine è troncata a destra. Nel caso in cui nell'immagine vi sia la lettera C ed il numero di caratteri del campo sia superiore a quello della stringa, ma la differenza tra le due lunghezze sia un numero dispari pari a 2n+1, la stringa è fatta precedere da n spazi e seguire da n+1 spazi. Se un campo immagine di stringa è composto dal solo apostrofo, è generato il primo carattere del valore stringa ad esso associato.

Note

1. L'istruzione immagine non è di tipo esecutivo.

2. I caratteri specificati al posto di un campo del tipo literal-field non possono essere:

   - l'apostrofo, se è seguito dalla lettera L od R o C

   - il punto, se è preceduto dal segno $ oppure dal segno # .

3. Come si vede dal formato delle istruzioni BUILD USING, DISP USING, PRINT USING e MAT PRINT USING, l'immagine di formato può essere assegnata ad una variabile stringa mediante l'istruzione di assegnazione:

   string-var = "immagine di formato"

4. Nel dimensionare un campo immagine di un numero, si ricordi cha al numero delle cifre previste per il numero deve essere aggiunto un carattere che occupi un posto per il segno algebrico.

5. Non si può specificare una istruzione immagine senza specificare almeno un campo immagine.

Esempi

1. Vediamo come si può utilizzare l'istruzione immagine in un programma per stampare dei valori numerici. La stampa effettuata con l'istruzione 90 mostra come nell'istruzione immagine vi possono essere dei caratteri che sono stampati esattamente come specificati. Il numero intero -5 è stampato allineato a destra nel relativo campo della istruzione immagine specificata. Il numero positivo 6 è preceduto da uno spazio. L'istruzione 100 stampa cinque asterischi perchè il valore 12345, considerando il segno + (implicito), è maggiore del numero di posizioni del campo ad esso relativo nella istruzione immagine 60 (5 posizioni). L'istruzione 110 stampa i valori in essa specificati su altrettante linee di stampa, tra loro distinte, perchè l'istruzione immagine relativa contiene un solo campo. L'istruzione 120 stampa due valori troncati delle relative cifre decimali, perchè il campo numerico specificato è un campo immagine di numero intero. L'istruzione 130 stampa i valori specificati allineati,

nel relativo campo immagine, rispetto al punto decimale. Dopo il punto decimale sono stampati degli zeri nelle posizioni previste dal campo, se il numero ha meno cifre decimali di quelle previste. Un valore nel formato esponenziale viene trasformato nel formato decimale corrispondente (vedi il valore 12 E 10). L'istruzione 150 si riferisce ad una immagine di stampa che è stata assegnata ad una variabile stringa; si osservi come il valore specificato (-25.8) è arrotondato e non troncato nella parte decimale perchè il campo immagine relativo è del tipo immagine di numero decimale (infatti è stato aggiunto un punto dopo i segni di numero). Infine le istruzioni 170 e 190 si riferiscono rispettivamente a campi immagine di numero esponenziale e con segno di $.

```
LIST
FILE    +IMAGE


0010 PRINT "Per poter controllare le posizioni relitive ai diversi campi della"
0020 PRINT "istruzione IMMAGINE si osservi la seguente stampa."
0030 PRINT
0040PRINT"123456789012345678901234567890123456789012345678901234567890123456789"
0050 :##########################   primo valore      ##   secondo valore.
0060 :     #####
0080 :################.###
0090 PRINT USING 50,-5,6
0100 PRINT USING 60,12345
0110 PRINT USING 60,0001,0002,0003
0120 PRINT USING 60,123.4,123.75
0130 PRINT USING 80,1.1,12E10
0140 LET A$="###."
0150 PRINT USING A$,-25.8
0160 :#####.##########################↑↑↑↑
0170 PRINT USING 160,-12,12345678901E77,-123,1234567890
0180 :$$$$$$$$$$$$$     $######.####       $$$$.###↑↑↑↑
0190 PRINT USING 180,1500,25.45,25.15E15
0200 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
Per poter controllare le posizioni relitive ai diversi campi della
istruzione IMMAGINE si osservi la seguente stampa.

123456789012345678901234567890123456789012345678901234567890123456789
                        -5   primo valore       6   secondo valore.
     *****
         1
         2
         3
       123
       123
              1.100
   120000000000.000
-26.
-1200.000000000000000000E-02
 1234.567890100000000000E+84
-1230.000000000000000000E-01
 1234.567890000000000000E+06
        $1500     $   25.4500       $251.500E+14
```

2. Vediamo l'impiego della istruzione immagine in un programma per la stampa di stringhe. Come si vede nel programma sottostante, le istruzioni immagine di formato possono essere poste anche all'interno di un ciclo FOR/NEXT (infatti sono istruzioni non esecutive). Le istruzioni 150, 160 e 170 stampano i relativi valori allineati a sinistra, a destra ed al centro rispetto al relativo campo immagine. L'istruzione 180 stampa un valore stringa ed un valore numerico oltre che i caratteri specificati come "literal-field". L'istruzione 190 prosegue la stampa nella linea successiva con la stessa immagine di stampa, perchè i valori specificati superano in numero i relativi campi immagine. Infine dalla stampa relativa all'istruzione 200 si vede che se si specificano meno valori di campi immagine i campi immagine in eccedenza sono ignorati.

```
LIST
FILE     MNB

0010 PRINT
0020 PRINT "Per poter controllare le posizione relative ai diversi campi della"
0030 PRINT "istruzione IMMAGINE si osservi la seguente stampa."
0040 PRINT
0050PRINT"123456789012345678901234567890123456789012345678901234567890123456789"
0060 FOR I=1 TO 3 STEP 1
0070 PRINT
0080 :'LLLLLLLLLLLL
0090 :'RRRRRRRRRRRRR
0100 :'CCCCCCCCCCCCC
0110 :'LLLLLLL  ↑↑↑↑    $$$$    !"#$%&'()_=+ALCRVXYasdfgla
0120 PRINT I
0130 NEXT I
0140 :'LLLLLLLL     ::::    'CCCCCCC    'RRRRRRRR
0150 PRINT USING 80,"OLIVETTI","P6066"
0160 PRINT USING 90,"Olivetti","P6066"
0170 PRINT USING 100,"OLIVETTI","P6066"
0180 PRINT USING 110,"Marzo",125
0190 PRINT USING 140,"Maggio","Giugno","Luglio","Agosto"
0200 PRINT USING 140,"FINE"
0210 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

Per poter controllare le posizione relative ai diversi campi della
istruzione IMMAGINE si osservi la seguente stampa.

123456789012345678901234567890123456789012345678901234567890123456789

 1

 2

 3
```

```
OLIVETTI
P6066.
      Olivetti
       __.P6066'
   OLIVETTI
    P6060
Marzo      ↑↑↑↑     $125    !"#$%&'()_=+ALCRVXYasdfgl@
Maggio'          ::::      Giugno          Luglio
Agosto           ::::
FINE             ::::
```

Istruzione INPUT

Funzione

Assegna i valori introdotti da tastiera alle variabili di programma specificate.

Formato

INPUT {**num-var** | **string-var**} [, {**num-var** | **string-var**}] ...

dove:

num-var
è una variabile numerica, semplice o con indice, a cui è assegnato il valore numerico introdotto da tastiera

string-var
è una variabile stringa, semplice o con indice, a cui è assegnata una stringa di caratteri introdotta da tastiera

Azione

L'esecuzione del programma si ferma e sul display è visualizzato un punto interrogativo. L'utente può introdurre dei valori, separati da virgola, che sono assegnati nell'ordine con cui sono introdotti alle variabili presenti nella istruzione INPUT. Dopo aver introdotto tutti i valori richiesti dalla istruzione, premendo [END OF LINE] l'esecuzione del programma continua dall'istruzione esecutiva successiva.

Note

1. Se l'operatore introduce meno dati di quelli richiesti dalla istruzione INPUT sul display appare ?? ed il sistema attende l'introduzione dei restanti dati.

2. I dati introdotti da tastiera devono essere coerenti con il tipo di variabile cui sono assegnati, ma si può assegnare un numero ad una variabile stringa.

3. Se si introducono più dati di quanti richiesti dall'istruzione INPUT i dati in eccesso sono ignorati e l'esecuzione del programma continua. Sul display appare il messaggio TOO MUCH INPUT-EXCESS IGNORED.

4. Una stringa contenente la virgola, o spazi iniziali e finali, deve essere introdotta da tastiera tra virgolette.

5. Non si può introdurre il carattere virgolette (esso può essere introdotto con l'impiego dell'istruzione RKB).

6. E' opportuno far precedere l'istruzione INPUT da una istruzione DISP o PRINT per indicare quali dati devono essere introdotti.

7. Se una virgola è posta all'interno di un dato essa provoca l'assegnazione della prima parte del dato alla variabile ad esso associata e lo spostamento di tutte le assegnazioni successive.

8. Se in un programma vi è la seguente sequenza di istruzioni:

```
140   I  =  10
150  INPUT I, B (I)
```

   allora l'indice della variabile con indice è 10 e non il nuovo valore assegnato ad I da tastiera.

9. Se viene assegnata una stringa ad una variabile numerica, viene annullata solamente la linea contenente il dato errato e sul display appare il messaggio:

   INCORRECT FORMAT - RETYPE LINE

   L'operatore deve quindi reintrodurre l'intera linea. Anche nel caso in cui si assegna un valore numerico, con esponente in valore assoluto maggiore di 63, ad una variabile dichiarata in singola precisione, compare sul display il messaggio suddetto. L'operatore deve quindi reintrodurre l'intera linea. Infine, lo stesso messaggio appare sul display se tra un dato ed il successivo sono introdotte da tastiera due o più virgole; l'operatore può correggere la linea introdotta ripetendo l'introduzione dei dati.

10. Se si assegna ad una variabile stringa una stringa con un numero di caratteri maggiore di quello dichiarato per la variabile suddetta, è segnalato un errore ed il sistema commuta nello stato di debugging assegnando alla variabile la stringa introdotta troncata a destra dei caratteri eccedenti la lunghezza di allocazione della variabile stringa.

11. Quando il sistema è in attesa di dati da tastiera si può premere il tasto STEP per commutare il sistema nello stato di debugging. Premendo una seconda volta il tasto STEP oppure il tasto CONTINUE sul display riappare il punto interrogativo ed il sistema è sempre in attesa dell'introduzione del dato precedente. Se si era premuto CONTINUE dopo che i dati sono introdotti ed assegnati alle variabili l'esecuzione del programma continua, mentre se si era premuto STEP il sistema ricommuta nello stato di debugging e sul display appare il numero di linea della istruzione che sarà eseguita successivamente.

Esempi

1. Eseguendo cinque volte la routine sottostante si possono mettere in luce le seguenti situazioni. Si può assegnare ad una variabile stringa un valore numerico senza che sia incluso tra virgolette (vedi il numero 12 assegnato ad A$). Si può assegnare ad una variabile stringa una stringa contenente la virgola se la stringa è compresa tra virgolette (vedi AREA, PESO). Se non si introducono tutti i dati previsti da una istruzione INPUT, sul display appare la richiesta di attesa di altri dati da tastiera (??). Durante la terza esecuzione della istruzione INPUT si sono introdotti più dati di quelli richiesti (A,1,2,B) per cui l'esecuzione continua ed il sistema ignora i dati in eccedenza (B) visualizzando sul display il messaggio TOO MUCH INPUT-EXCESS IGNORED. Durante la quarta esecuzione della istruzione INPUT viene introdotto un valore (V) non coerente con il tipo di variabile per cui il sistema visualizza il messaggio INCORRECT FORMAT-RETYPE LINE ed attende una nuova introduzione. Durante la quinta esecuzione della istruzione INPUT si introducono due virgole di seguito per cui viene visualizzato di nuovo il messaggio suddetto. L'intera linea deve essere digitata di nuovo. Come terzo dato si è introdotta una stringa di 19 caratteri

per cui il sistema visualizza l'errore ERROR 8 IN LINE 20. Il sistema è nello stato di debugging (la luce del tasto STEP è accesa). Premendo CONTINUE l'esecuzione del programma viene portata a termine e la stringa è troncata dopo i primi 16 caratteri (B$ infatti ha una lunghezza di allocazione di 16 caratteri).

```
LIST
FILE    +INPUT


0010 LET B=1
0020 INPUT A$,A,B$
0030 PRINT "A$=";A$,"A=";A,"B$=";B$
0040 LET B=B+1
0050 IF B<=5 THEN 20
0060 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
?
12,12,"AREA,PESO"
A$=12            A= 12            B$=AREA,PESO
?
!!!!!!
??
126
??
######
A$=!!!!!!        A= 126           B$=######
?
A,1,2,B
TOO MUCH INPUT-EXCESS IGNORED
A$=A             A= 1             B$=2
?
F
??
U
INCORRECT FORMAT-RETYPE LINE
5,S
A$=F             A= 5             B$=S
?
Q,,1
INCORRECT FORMAT-RETYPE LINE
Q,1,1234567890123456789
ERROR 8   IN LINE 20
A$=Q             A= 1             B$=1234567890123456
```

2. Nella routine sottostante vi è una istruzione INPUT in cui compare una variabile che è utilizzata anche come indice di una successiva variabile con indice. Come si vede la variabile B (2) mantiene il valore precedente mentre il valore introdotto da tastiera (90) è assegnato alla variabile B (10).

```
LIST
FILE     +INPUT1

0010 DISP "Introduci i dati"
0020 LET B(2)=8
0030 LET I=10
0040 INPUT I,B(I)
0050 PRINT "I=";I,"B(10)=";B(10),"B(2)=";B(2)
0060 END

END OF LISTING


RUN
Introduci i dati
?
2,90
I= 2            B(10)= 90        B(2)= 8
```

3. Nell'esempio seguente si mostrano due esecuzioni della routine sottostante. Durante la prima esecuzione, quando viene eseguita l'istruzione INPUT si preme il tasto STEP. Il sistema è nello stato di debugging. Premendo il tasto STEP di nuovo sul display riappare il punto interrogativo e introducendo il valore l'istruzione INPUT viene portata a termine; premendo successivamente STEP il programma è eseguito passo a passo. Durante la seconda esecuzione, quando viene eseguita l'istruzione INPUT si preme il tasto STEP. Il sistema è nello stato di debugging. Questa volta si preme il tasto CONTINUE e sul display appare di nuovo il punto interrogativo. Introducendo il dato richiesto l'esecuzione del programma viene portata a termine.

```
LIST
FILE

0010 DCL S A
0020 INPUT A
0030 PRINT "A=";A
0040 END

END OF LISTING
```

```
RUN
#20
?
STEP        IN LINE 20
#20
?
124
STEP        IN LINE 30
#30
A= 124
STEP        IN LINE 40
#40
```

```
RUN
#20
?
STEP        IN LINE 20
#20
?
124
#30
A= 124
#40
```

Istruzione
INTERRUPT ENABLE

Funzione

Consente la gestione da parte dell'utente di interruzioni interne ed esterne.

Formati

1. INTERRUPT ENABLE (I,funam)
2. INTERRUPT ENABLE (E,funam,intmsk,priort)

dove:

I
specifica che le interruzioni interne, cioè errori rilevati durante l'esecuzione, saranno gestiti dall'utente

E
specifica che le interruzioni esterne, cioè quelle generate da unità periferiche esterne saranno gestite dall'utente

Per maggiori informazioni sulla gestione di interruzioni esterne, cioè il formato qui contraddistinto dal numero 2, si faccia riferimento al manuale "P6066 I/O con periferiche esterne"

funam
è una costante stringa o una variabile stringa il cui valore può essere o un carattere alfabetico maiuscolo o un asterisco. Il carattere alfabetico specifica il nome di una definizione di funzione numerica multilinea.

Azione

L'esecuzione dell'istruzione con funam che ha per valore un carattere alfabetico fa sì che ogni successivo errore di sistema passi il controllo alla funzione multilinea richiamata dall'istruzione.

L'esecuzione dell'istruzione con funam che ha per va-

lore un asterisco fa sì che ogni successivo errore sia gestito dal sistema.

Note

1. L'azione richiesta dall'istruzione INTERRUPT ENABLE resta valida fino ad una successiva istruzione INTERRUPT ENABLE eseguita.

2. L'istruzione DEF della funzione multilinea deve avere come parametri due variabili numeriche semplici. Alla prima variabile viene assegnato come valore il codice d'errore che ha causato l'interruzione; alla seconda variabile viene assegnato come valore il numero di linea dell'istruzione in cui si è verificato l'errore. L'utente può quindi analizzare questi valori per determinare quale azione intraprendere.

3. La funzione multilinea deve assegnare un valore alla pseudo variabile FN*. Se il valore assegnato è 0, l'utente potrà gestire l'errore. Se il valore assegnato è diverso da 0, l'utente dovrà accettare l'azione intrapresa dal Sistema. In quest'ultimo caso, l'esecuzione del programma resta sospesa e ci sarà il messaggio:

   ERROR cod-errore IN LINE num. linea

   Si potrà poi decidere quale azione intraprendere a seconda se l'errore sarà o no recuperabile.

4. Se FN*= 0 e l'errore è recuperabile, l'esecuzione riprenderà, a completamento della funzione multilinea, con la prima istruzione eseguibile che segue l'istruzione che ha causato l'errore.

5. Se FN*= 0 e l'errore non è recuperabile, l'esecuzione riprenderà, a completamento della funzione multilinea, con l'istruzione che ha causato l'errore.

6. L'istruzione sarà ignorata nel caso dei seguenti errori non recuperabili:

   - 65 - non c'è spazio sufficiente in memoria utente per continuare l'esecuzione

   - 74 - è stato superato il numero massimo di riferimenti ad altre definizioni di funzione mono-

linea o multilinea all'interno di una definizione monolinea o multilinea (255).

7. Se insorge un errore durante l'esecuzione della funzione multilinea chiamata dall'istruzione INTERRUPT ENABLE, non è possibile apportare correzioni. Il sistema invierà il messaggio d'errore 282.

8. Vedere l'appendice D per l'elenco completo degli errori.

Esempi

1. L'esempio che segue mostra l'istruzione INTERRUPT ENABLE utilizzata per gestire una condizione di underflow (codice d'errore 4). L'underflow avviene al passo 96 del ciclo FOR/NEXT, ma l'esecuzione del programma non viene sospesa. Dopo aver cambiato il valore di A alla linea 90 e dopo che il controllo è ritornato all'istruzione successiva a quella che ha causato l'errore, il programma va avanti fino alla fine, senza che il sistema commuti in stato di debugging, come succederebbe se non si fosse utilizzata l'istruzione INTERRUPT ENABLE e la routine di gestione errori.

```
FILE     UKERR4

0005 INTERRUPT ENABLE (I,"A")
0010 LET L=100
0015 LET A=.0001
0020 FOR I=1 TO L STEP 1
0030 LET A=A/10
0040 NEXT I
0050 GOTO 140
0060 DEF FNA(C,L)
0070 IF C<>4 THEN 110
0080 PRINT "SI E' VERIFICATO UNDERFLOW IN LINEA ";L;"AL PASSO";I
0090 LET A=1E5
0095 LET FN*=0
0100 GOTO 130
0110 PRINT "SI E' VERIFICATO ERRORE ";C;" IN LINEA ";L
0120 LET FN*=1
0130 FNEND
0140 PRINT "A =",A
0150 END

END OF LISTING


SI E' VERIFICATO UNDERFLOW IN LINEA  30 AL PASSO 96
A = 10
```

2. In questo esempio è stata eliminata la linea 15 del programma precedente. Il primo errore che si incontra è il codice di errore 1 "non è stato assegnato alcun valore ad una variabile numerica". L'istruzione PRINT alla linea 110 tiene conto dell'errore. L'istruzione FN* della linea 120 richiede poi la normale gestione degli errori da parte del sistema. Come risultato si avrà la sospensione dell'esecuzione del programma e il sistema che commuterà in stato di debugging.

```
FILE     UKERR4

0005 INTERRUPT ENABLE (I,"A")
0010 LET L=100
0020 FOR I=1 TO L STEP 1
0030 LET A=A/10
0040 NEXT I
0050 GOTO 140
0060 DEF FNA(C,L)
0070 IF C<>4 THEN 110
0080 PRINT "SI E' VERIFICATO UNDERFLOW IN LINEA ";L;"AL PASSO";I
0090 LET A=1E5
0095 LET FN*=0
0100 GOTO 130
0110 PRINT "SI E' VERIFICATO ERRORE ";C;" IN LINEA ";L
0120 LET FN*=1
0130 FNEND
0140 PRINT "A =";A
0150 END

END OF LISTING



SI E' VERIFICATO ERRORE  1  IN LINEA  30
```

Istruzione LET

Funzione

Assegna i valori alle variabili di programma.

Formato

[LET] { **num-var [= num-var] ... = num-exp** / **string-var [= string-var] ... = string-exp** }

dove:

num-var
è una variabile numerica (semplice o con indice) a cui viene assegnato il valore della espressione numerica specificata alla destra dell'ultimo segno uguale (ultimo da sinistra)

num-exp
è una espressione numerica che specifica il valore da assegnare alla variabile od alle variabili indicate alla sinistra dell'ultimo segno uguale

string-var
è una variabile stringa (semplice o con indice) a cui viene assegnato il valore della espressione stringa specificata alla destra dell'ultimo segno uguale

string-exp
è una espressione stringa che specifica il valore da assegnare alla variabile od alle variabili indicate alla sinistra dell'ultimo segno uguale

Azione

L'espressione a destra dell'ultimo segno uguale è eseguita ed il valore ottenuto è assegnato alle variabili indicate alla sinistra dello stesso segno.

Note

1. La parola chiave LET è opzionale.

2. Il numero di variabili a sinistra del segno di assegnazione è limitato solamente dal numero massimo di caratteri che possono comporre una linea (80).

3. La lunghezza attuale di una variabile stringa presente in una istruzione di assegnazione diventa uguale al numero di caratteri che costituiscono la stringa risultato della espressione stringa indicata.

4. Se la lunghezza di allocazione di una variabile stringa è inferiore al numero di caratteri della stringa risultante dalla esecuzione della espressione stringa, quest'ultima viene troncata dei caratteri eccedenti. Il sistema commuta nello stato di debugging e viene segnalato un errore di tipo recuperabile.

5. L'indice di una variabile con indice che compare in una istruzione LET può essere espresso, in generale, mediante una espressione numerica. Se l'espressione suddetta contiene delle variabili i cui valori sono modificati durante l'esecuzione della istruzione LET, l'espressione viene valutata con i valori che le variabili avevano prima della esecuzione della istruzione LET (vedi l'esempio più avanti).

6. Se ad una variabile numerica in singola precisione viene assegnato un valore nella zona di OVERFLOW per la singola precisione, la variabile suddetta assume il valore 9.99999E63 oppure -9.99999E63. Il sistema è nello stato di debugging e viene visualizzato un errore di tipo recuperabile.

7. Se il risultato del calcolo di una espressione numerica è un valore nella zona di OVERFLOW per la doppia precisione, alla variabile in doppia precisione, specificata alla sinistra del segno uguale, viene assegnato il valore 9.999999999999E99 oppure -9.999999999999E99. Il sistema è nello stato di debugging ed è visualizzato un errore di tipo recuperabile.

8. Se ad una variabile numerica è assegnato un valore nella zona diUNDERFLOW (relativa al tipo di precisione con cui è dichiarata) allora la variabile suddetta assume il valore zero. Il sistema è nello stato di debugging e viene visualizzato un errore di tipo recuperabile.

9. Se nella espressione numerica specificata con num-exp compare una variabile a cui non è ancora stato

assegnato un valore, l'espressione è valutata assegnando alla variabile suddetta il valore zero. Il sistema commuta nello stato di debugging e viene visualizzato un messaggio di errore di tipo recuperabile.

10. Se nella espressione stringa specificata con string-exp compare una variabile a cui non è stato ancora assegnato un valore, l'espressione è valutata assegnando alla variabile suddetta il valore di stringa nulla. Il sistema commuta nello stato di debugging e viene visualizzato un messaggio di errore di tipo recuperabile.

Esempi

1. Nella routine sottostante possiamo notare che con una istruzione LET si può assegnare contemporaneamente lo stesso valore a 33 variabili. La variabile A è quindi utilizzata come un contatore. Infine il valore della variabile A1 è modificato assegnandole il valore della espressione specificata a destra del segno uguale nella istruzione 80.

```
LIST
FILE    +LET

0010A=B=C=D=E=F=G=H=G=J=K=L=M=N=O=P=Q=R=S=T=U=V=W=Z=A0=A1=A2=A3=A4=A5=A6=A7=A8=4
0020 PRINT A,B,C,D,G
0030 LET A=0
0040 LET A=A+1
0050 PRINT A
0060 IF A<=4 THEN 40
0070 PRINT "A1=";A1
0080 LET A1=PI*A5+SQR(P*Q+R-5)/LGT(W*Z)
0090 PRINT "A1=";A1
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 4              4              4              4              4
 1
 2
 3
 4
 5
A1= 4
A1= 15.782814
```

2. Vediamo alcuni esempi di impiego della istruzione LET per assegnare un valore ad una variabile stringa. Si veda, istruzioni 150 ÷ 180, come si possono aggiungere via via dei caratteri al valore di una variabile stringa.

```
LIST
FILE    *LET1

0010 PRINT
0020 PRINT
0030 LET A$="I capitoli"
0040 LET B$=" 1 "
0050 LET C$=" 2 "
0060 LET D$=" 3 "
0070 LET E$=" 4 e 5 "
0080 DCL 80 (F$,G$,H$)
0090 LET F$=" descrivono il sistema."
0100 LET G$="descrivono il linguaggio."
0110 LET H$=A$+B$+","+C$+"e"+D$+F$
0120 PRINT "H$=";H$
0130 LET H$=A$+E$+G$
0140 PRINT "H$=";H$
0150 LET S$=""
0160 FOR I=0 TO 9 STEP 1
0170 LET S$=S$+CHR$(I)
0180 NEXT I
0190 PRINT "S$=";S$
0200 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****


H$=I capitoli 1 , 2 e 3  descrivono il sistema.
H$=I capitoli 4 e 5 descrivono il linguaggio.
S$=■Γ⌊⌋≥■/o↖→
```

Istruzione NEXT

Funzione

Definisce il termine di un ciclo iterativo.

Formato

NEXT control-var

dove:
control-var
è la variabile semplice numerica specificata come variabile di controllo nella corrispondente istruzione FOR.

Azione

Vedi istruzione FOR.

Istruzione ON...GOSUB

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma scelto tra un insieme di sottoprogrammi in funzione del valore assunto da una espressione specificata.

Formato

ON num-exp GOSUB line-num [, line-num] ...

dove:

num-exp

indica un espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica quale sottoprogramma deve essere eseguito tra quelli che iniziano con l'istruzione il cui numero di linea è specificato dopo la parola chiave GOSUB

line-num

è il numero di linea della prima istruzione di un sottoprogramma

Azione

L'espressione numerica è eseguita ed il suo valore arrotondato all'intero più prossimo. Il controllo della esecuzione del progamma è trasferito all'istruzione il cui numero di linea si trova nella istruzione ON... GOSUB nella posizione (da sinistra a destra) indicata dal numero ottenuto come risultato della espressione numerica di cui sopra. Così una espressione il cui valore è 3.85, trasferisce il controllo alla istruzione il cui numero di linea è indicato al quarto posto nell'insieme dei numeri di linea che costituiscono gli operandi della istruzione ON...GOSUB. Ognuna delle istruzioni il cui numero di linea è indicato nella istruzione ON...GOSUB è la prima di un insieme di istruzioni BASIC che costituiscono altrettanti sottoprogrammi. L'ultima istruzione di ogni sottoprogramma è l'istruzione RETURN che trasferisce il controllo della esecuzione del progamma alla prima istruzione esecuti-

va (in ordine di numero di linea) successiva alla istruzione ON...GOSUB.

Note

1. In ogni sottoprogramma vi può essere più di una istruzione RETURN.

2. I sottoprogrammi possono far parte di una funzione multilinea: in questo caso anche l'istruzione ON...GOSUB deve essere un'istruzione della funzione multilinea.

3. Se il valore della espressione numerica, approssimato all'intero più prossimo, è minore di 1 o maggiore del numero "numeri di linea" presenti nella istruzione, l'esecuzione del programma prosegue dall'istruzione esecutiva successiva alla istruzione ON...GOSUB.

4. Una istruzione ON...GOSUB può essere contenuta in un ciclo FOR/NEXT, ma i sottoprogrammi non possono essere contenuti in un ciclo FOR/NEXT.

5. Per ulteriori note si veda l'istruzione GO SUB.

Esempio

Nel seguente programma sono riportati quattro sottoprogrammi a cui si fa riferimento mediante una istruzione ON...GOSUB. Si osservi, attraverso le diverse esecuzioni, come il controllo della esecuzione del programma dipenda essenzialmente dal valore della espressione A*B.

```
LIST
FILE    *ONGOS

0010 PRINT "Ecco un esempio di programma che utilizza piu` sottoprogrammi."
0020 LET A=1
0030 DISP "Scegli il sottoprogramma                          ";
0040 INPUT B
0044 PRINT
0046 PRINT "A*B=";A*B
0048 PRINT
0050 ON A*B GOSUB 100,200,300,400
0060 IF A=2 THEN 430
0070 PRINT "Non e` stato eseguito alcun sottoprogramma!"
0080 GOTO 430
0100 PRINT "E` stato eseguito il primo sottoprogramma!"
0110 LET A=2
0120 RETURN
0200 PRINT "E` stato eseguito il secondo sottoprogramma!"
0210 LET A=2
0220 RETURN
```

```
0300 PRINT "E` stato eseguito il terzo sottoprogramma!"
0310 LET A=2
0320 RETURN
0400 PRINT "E` stato eseguito il quarto sottoprogramma!"
0410 LET A=2
0420 RETURN
0430 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
-15

A*B=-15

Non e` stato eseguito alcun sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
1.45

A*B= 1.45

E` stato eseguito il primo sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
1.56

A*B= 1.56

E` stato eseguito il secondo sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
3

A*B= 3

E` stato eseguito il terzo sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
4

A*B= 4

E` stato eseguito il quarto sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
4.56

A*B= 4.56

Non e` stato eseguito alcun sottoprogramma!
```

Istruzione ON...GOTO

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad una istruzione scelta tra un insieme di istruzioni in funzione del valore assunto da una espressione specificata.

Formato

ON num-exp GOTO line-num [, **line-num**] ...

dove:

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica a quale istruzione di programma, tra quelle il cui numero di linea è specificato dopo la parola chiave GOTO, deve essere trasferito il controllo della esecuzione

line-num
indica il numero di linea di una istruzione del programma

Azione

L'espressione numerica è eseguita ed il suo valore arrotondato all'intero più prossimo. Il controllo della esecuzione del programma è trasferito alla istruzione il cui numero di linea si trova nella istruzione ON...GOTO nella posizione (da sinistra a destra) indicata dal numero ottenuto come risultato della espressione numerica di cui sopra. Così una espressione il cui valore è 2.75, trasferisce il controllo della esecuzione del programma alla istruzione il cui numero di linea è indicato al terzo posto nell'insieme dei numeri di linea che costituiscono gli operandi della istruzione ON...GOTO.

Note

1. L'istruzione ON...GOTO può trasferire il controllo della esecuzione del programma ad una istruzione di una funzione multilinea, se anche l'istruzione

ON...GOTO fa parte dell'insieme di istruzioni della funzione multilinea.

2. Se il valore della espressione approssimato all'intero più prossimo è minore di 1 o maggiore del numero di "numeri di linea" presenti nella istruzione, l'esecuzione del programma prosegue dalla istruzione esecutiva successiva alla istruzione ON...GOTO.

3. Se l'istruzione ON...GOTO è compresa in un ciclo FOR/NEXT anche le istruzioni i cui numeri di linea sono indicati nella istruzione stessa devono far parte del ciclo FOR/NEXT.

4. Se l'istruzione ON...GOTO è esterna ad un ciclo FOR/NEXT i numeri di linea specificati come operandi non possono appartenere ad istruzioni interne al ciclo suddetto.

Esempio

Il programma sottostante mostra come si può utilizzare l'istruzione ON...GOTO in un ciclo FOR/NEXT. Si hanno cinque esecuzioni della suddetta istruzione per ogni esecuzione completa del programma. Sono stati forniti, in successione, i valori 0, 1.23, 2., 2.8, 4 e 89 che permettono di esemplificare tutti i casi possibili per il controllo del corretto funzionamento del programma stesso.

```
LIST
FILE

0010 PRINT "Ecco un esempio di programma che utilizza l'istruzione ON...GOTO.
0015 FOR I=1 TO 6 STEP 1
0020 DISP "Scegli l'instruzione da eseguire.                ";
0030 INPUT A
0040 ON A GOTO 100,200,300,400
0050 PRINT "Hai scelto di terminare subito il programma."
0060 GOTO 420
0100 PRINT "Hai scelto di eseguire la prima istruzione del set!"
0110 GOTO 420
0200 PRINT "Hai scelto di eseguire la seconda istruzione del set!"
0210 GOTO 420
0300 PRINT "Hai scelto di eseguire la terza istruzione del set!"
0310 GOTO 420
0400 PRINT "Hai scelto di eseguire la quarta istruzione del set!"
0410 GOTO 420
0420 NEXT I
0450 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio di programma che utilizza l'istruzione ON...GOTO.
Scegli l'instruzione da eseguire.                ?
0
Hai scelto di terminare subito il programma.
Scegli l'instruzione da eseguire.                ?
1.23
Hai scelto di eseguire la prima istruzione del set!
Scegli l'instruzione da eseguire.                ?
2.
Hai scelto di eseguire la seconda istruzione del set!
Scegli l'instruzione da eseguire.                ?
2.8
Hai scelto di eseguire la terza istruzione del set!
Scegli l'instruzione da eseguire.                ?
4
Hai scelto di eseguire la quarta istruzione del set!
Scegli l'instruzione da eseguire.                ?
89
Hai scelto di terminare subito il programma.
```

Istruzione PAD

Funzione

Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione aggiungendo in coda dei caratteri predefiniti.

Formato

PAD string-var, n

dove:

string-var

è una variabile stringa alla quale sono aggiunti come caratteri di riempimento, fino ad eguagliare la dimensione attuale della variabile con quella di allocazione, i caratteri ISO corrispondenti al numero intero n

n

è un numero intero compreso tra Ø e 255 che indica quale carattere ISO deve essere utilizzato come carattere di riempimento

Azione

Alla variabile stringa string-var sono aggiunti caratteri di riempimento corrispondenti, secondo la tabella ISO (vedi appendice E), al numero n, finchè la sua lunghezza attuale eguagli la lunghezza di allocazione.

Nota

L'istruzione PAD permette la generazione di record di dati con lunghezza prefissata.

Esempi

1. Nell'esempio che segue si vede che l'istruzione PAD è una istruzione di assegnazione del carattere specificato alla variabile indicata. Se infatti alla variabile A$ non si assegna alcun valore, ad essa viene assegnata, istruzione 20, una stringa di tanti asterischi (valore corrispondente 42) quanto è la lunghezza di allocazione (in questo caso 16 caratteri).

```
LIST
FILE

0010 PAD A$,42
0011 PRINT
0012 PRINT
0013 PRINT
0020 PRINT "A$=";A$
0030 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****


A$=****************
```

2. Ecco un tipico esempio d'impiego dell'istruzione PAD per la creazione di record di eguale lunghezza da registrare su file dati esterno.

```
LIST
FILE    +PAD

0010 FILES A
0020 SCRATCH :1
0030 FOR I=1 TO 10 STEP 1
0040 DISP "Introduci il record
0050 INPUT A$
0060 PAD A$,42
0070 WRITE :1,A$
0080 NEXT I
0090 PRINT
0100 PRINT
0110 PRINT
0120 PRINT "Ecco i dati registrati nel file A;prima stampo tutto il record "
0130 PRINT "e poi solo il suo contenuto."
0140 PRINT
0150 PRINT
0160 RESTORE :1
0170 FOR I=1 TO 10 STEP 1
0180 READ :1,B$
0190 PRINT B$
0200 DEPAD B$,42
0210 PRINT B$
0220 NEXT I
0230 END

END OF LISTING
```

```
RUN
Introduci il record                ;
?
Mario
Introduci il record                ;
?
Piero
Introduci il record                ;
?
Giovanni
Introduci il record                ;
?
Giacomo
Introduci il record                ;
?
Nicola
Introduci il record                ;
?
Enzo
Introduci il record                ;
?
Giulio
Introduci il record                ;
?
Enrico
Introduci il record                ;
?
Sandro
Introduci il record                ;
?
Carlo
```

Ecco i dati registrati nel file A;prima stampo tutto il record e poi solo il suo contenuto.

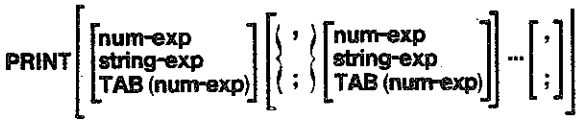
```
Mario***********
Mario
Piero***********
Piero
Giovanni********
Giovanni
Giacomo*********
Giacomo
Nicola**********
Nicola
Enzo************
Enzo
Giulio**********
Giulio
Enrico**********
Enrico
Sandro**********
Sandro
Carlo***********
Carlo
```

Istruzione PRINT

Funzione

Stampa dati e testi in un formato standard.

Formato



dove:

num-exp
indica una espressione numerica il cui valore deve essere stampato

string-exp
indica una espressione stringa il cui valore deve essere stampato

TAB (num-exp)
è una funzione di sistema che indica la posizione, corrispondente al valore di num-exp arrotondato all'intero più prossimo, in cui si deve posizionare il pointer del buffer di stampa

,
indica uno spostamento standard per il pointer del buffer di stampa

;
indica che il pointer del buffer di stampa deve rimanere nella posizione in cui si trova

Azione

I valori delle espressioni specificate nella istruzione sono stampati, nell'ordine con cui compaiono nella istruzione, con il formato descritto nelle note successive.

La posizione nella linea di stampa dei caratteri che sono stampati è controllata mediante l'impiego della virgola (,) del punto e virgola (;) e della funzione TAB, come specificato nel paragrafo "Controllo della posizione dei caratteri nel buffer di stampa".

Note

1. Il valore di una espressione stringa viene stampato con la sequenza dei caratteri che la compongono.

2. Il valore di una espressione numerica viene stampato anteponendo ad esso uno spazio (se è un numero positivo) od il segno meno (se è un numero negativo) ed aggiungendo in coda ad esso un altro spazio.

3. I numeri interi sono stampati con al massimo 8 cifre significative.

4. I numeri con valore assoluto compreso tra 0.0999999995 e 99999999.4 sono stampati, con al massimo 8 cifre significative (se il numero è minore di 1 viene tralasciato lo zero che precede la parte decimale), nel formato in virgola fissa.

5. I numeri con valore assoluto minore di 0.0999999995 che possono essere rappresentati con 8 cifre significative, sono stampati nel formato in virgola fissa.

6. Tutti gli altri numeri sono stampati nel formato in virgola mobile.

7. Le costanti stringa devono essere comprese tra virgolette.

8. Virgola e punto e virgola sono elementi separatori e finali. Sono obbligatori come elementi separatori fra le espressioni specificate come operandi. Sono opzionali come elementi finali.

9. Una istruzione PRINT senza operandi stampa il contenuto del buffer di stampa (vedi buffer di stampa più avanti) e pone il pointer nella prima posizione. Se il buffer non contiene caratteri l'effetto prodotto è una interlinea.

10. Se la configurazione di sistema installata è priva di stampante integrata e di stampante ausiliaria (vedi comando CONFIGURE), l'istruzione PRINT visualizza sul display le informazioni relative alle espressioni in essa specificate. Ad ogni visualizzazione l'esecuzione del programma si interrompe e riprende premendo il tasto CONTINUE. Per leggere le informazioni prodotte in una linea si devono utilizzare i tasti [→], [REPEAT] e [SHIFT] come spiegato nel

capitolo 1, par. "La tastiera".
Se le informazioni prodotte occupano più di una linea, le informazioni relative ad ogni linea successiva alla prima sono visualizzate premendo ogni volta CONTINUE. Quando, premendo il tasto CONTINUE, il sistema emette una segnale acustico, l'ultima linea visualizzata è la fine del testo prodotto dall'istruzione PRINT.

Controllo della posizione dei caratteri nel buffer di stampa

L'esecuzione di una istruzione PRINT genera in un registro di 80 caratteri, detto buffer di stampa (vedi figura 5-2), la stringa di caratteri corrispondenti al valore dell' espressione che compare nella istruzione stessa. Un secondo registro, detto pointer, indica la posizione del buffer in cui deve iniziare la stringa di caratteri suddetta. Ogni volta che un carattere è generato nel buffer di stampa, il pointer avanza di una posizione. Quando il buffer di stampa è pieno, il suo contenuto viene stampato sulla riga di stampa ed il pointer è rimesso ad 1 (indica la prima posizione del buffer).

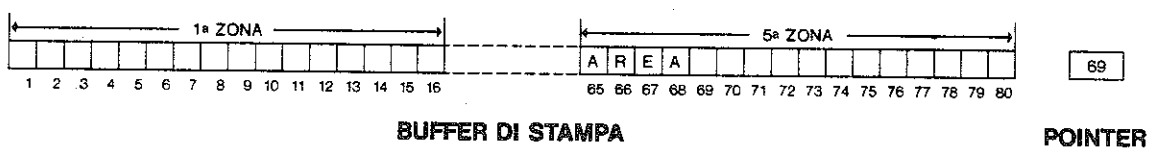


Figura 5-2 Buffer di stampa e relativo pointer

Ogni volta che è eseguita una istruzione PRINT senza virgola o punto e virgola come elementi finali, il contenuto del buffer di stampa è stampato ed il pointer di stampa è rimesso ad 1.

L'impiego della funzione TAB (num-exp) permette il controllo della posizione dei caratteri nel buffer di stampa. Infatti il valore dell'espressione numerica num-exp è arrotondato all'intero più prossimo e assegnato al pointer del buffer di stampa. Se il valore suddetto è minore di uno viene segnalato un errore.

Se il valore suddetto è maggiore di 80, è ridotto modulo 80; ossia è posto uguale ad n-80*INT((n-1)/80), dove n è il valore, arrotondato all'intero più prossimo, della espressione numerica num-exp. Se il pointer del buffer di stampa indica una posizione superiore al valore ritornato dalla funzione TAB, il contenuto del buffer è stampato ed il pointer indica la posizione ad esso assegnata in seguito alla valutazione di TAB. Per essere sicuri che i caratteri successivi siano generati a partire dalla posizione assegnata al pointer da TAB (num-exp) è bene far seguire tale funzione da un punto e virgola (;).

L'impiego della virgola (,) permette la stampa dei dati a distanza standard, di 16 posizioni, l'una dall'altra: il buffer di stampa è diviso in 5 zone, ognuna di 16 posizioni (vedi figura 5-2). In un programma con le seguenti istruzioni:

```
50 PRINT ,
60 PRINT "A"
```

l'istruzione 50 pone il pointer del buffer di stampa in posizione 17 per cui la successiva istruzione genera il carattere A nella posizione 17 del buffer medesimo. In un programma con le seguenti istruzioni:

```
50 PRINT "A", "B", "C", "D"
60 PRINT "E",
70 PRINT "F"
```

l'istruzione 50 genera nel buffer di stampa il carattere A nella prima posizione, il carattere B nella 17-esima posizione, il carattere C nella 33-esima posizione, il carattere D nella 49-esima posizione e li stampa nelle posizioni suddette. L'istruzione 60 genera il carattere E nella prima posizione del buffer di stampa. Il pointer indirizza la 17-esima posizione del buffer di stampa, per cui la successiva istruzione PRINT genera il carattere F nella posizione suddetta. Si noti che se, quando è eseguita una istruzione PRINT riferita ad una virgola, il pointer di stampa indica una posizione nella 5ª zona, il contenuto del buffer di stampa è stampato ed il pointer indirizza la posizione uno.

L'impiego del punto e virgola (;) permette di accodare nel buffer di stampa due o più valori corrispondenti ad espressioni distinte. Se in un programma si hanno le istruzioni:

```
20 PRINT "A"; "B"
30 PRINT "C";
40 PRINT "D"
```

l'istruzione 20 genera i caratteri A e B uno di seguito all'altro. Le istruzioni 30 e 40 producono lo stesso effetto per C e per D.

Si osservi che se il valore corrispondente ad una espressione specificata in una istruzione PRINT è tale per cui il numero dei caratteri ad esso corrispondente è superiore al numero di posizioni rimaste libere nel buffer di stampa, il contenuto del buffer di stampa è stampato ed i caratteri suddetti sono generati dall'inizio del buffer. Se la stringa suddetta è composta da più di 80 caratteri è stampata su più linee andando a campo ogni 80 caratteri.

Nel seguito diamo una tabella che riassume l'impiego della virgola e del punto e virgola come elementi separatori in una istruzione di tipo PRINT. Il separatore menzionato segue il dato, specificato nella prima colonna, nella istruzione PRINT relativa.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di stampa	Valore del pointer dopo la generazione dei caratteri nel buffer di stampa
Espressione numerica	","	Il valore corrispondente all'espressione è generato nel buffer di stampa dalla posizione indicata dal pointer, se le posizioni libere nel buffer sono sufficienti a contenerlo, altrimenti il contenuto del buffer è stampato ed il valore è generato nel buffer a partire dalla prima posizione.	Il valore del pointer avanza delle rimanenti posizioni della zona di stampa. Se viene raggiunta la fine del buffer di stampa, il contenuto del buffer è stampato ed il valore del pointer è 1.
	";" punto e virgola		Il pointer indica la posizione successiva all'ultima posizione occupata dal valore generato nel buffer.
Espressione stringa	"," virgola	La stringa corrispondente all'espressione è generata nel buffer di stampa dalla posizione indicata dal pointer, se le posizioni libere nel buffer sono sufficienti a contenerla, altrimenti il contenuto del buffer è stampato e la stringa è generata nel buffer a partire dalla prima posizione. Se la stringa ha più di 80 caratteri, ogni volta che il buffer è pieno, il suo contenuto è stampato ed i rimanenti caratteri della stringa sono generati dall'inizio del buffer.	Il valore del pointer avanza delle rimanenti posizioni della zona di stampa. Se viene raggiunta la fine del buffer di stampa, il contenuto del buffer è stampato ed il valore del pointer è 1.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di stampa	Valore del pointer dopo la generazione dei caratteri nel buffer di stampa
Espressione stringa	";" punto e virgola		Il pointer indica la posizione successiva all'ultima posizione occupata dalla stringa generata nel buffer.
stringa nulla	"," virgola	Non viene generato alcun carattere nel buffer di stampa.	Il valore del pointer avanza delle rimanenti posizioni della zona di stampa. Se viene raggiunta la fine del buffer, il contenuto del buffer di stampa è stampato ed il valore del pointer è 1.
	";" punto e virgola		Il valore del pointer non è modificato.

Tabella 5-2 Impiego della virgola e del punto e virgola con PRINT

Esempi

1. Nella routine seguente si può notare la funzione di tabulatore standard della virgola (istruzione 70), come vengono accodati i dati con il punto e virgola ed infine come la istruzione END fa stampare il contenuto del buffer di stampa (quando è eseguita l'istruzione DISP i dati sono ancora nel buffer di stampa; è l'istruzione END, in questo caso, che li trasmette alla stampante).

```
LIST
FILE    *PRINT


0010 PRINT
0020 PRINT
0030 PRINT "Per poter controllare la posizione dei caratteri nella linea, "
0040 PRINT " si osservi la seguente stampa:"
0050 PRINT "12345678901234567890123456789012345678901234567890123456789012345678"
0060 PRINT
0070 PRINT ,,123456789,,12
0080 PRINT 45E88;45E88;45E88;-45E88;-45E88
0090 PRINT -89E-78;-34E-56;
0100 DISP "I dati della istruzione 90 sono ancora nel buffer di stampa!"
0110 DELAY 300
0120 END

END OF LISTING
```

```
RUN

Per poter controllare la posizione dei caratteri nella linea,
 si osservi la seguente stampa:
12345678901234567890123456789012345678901234567890123456789012345678901234567

                                    1.2345679E+08                    12
 4.5000000E+89  4.5000000E+89  4.5000000E+89 -4.5000000E+89 -4.5000000E+89
I dati della istruzione 90 sono ancora nel buffer di stampa!
-8.9000000E-77 -3.4000000E-55
```

2. Vediamo come i numeri stampati in virgola mobile sono distanziati diversamente se separati da virgola o da punto e virgola. Vediamo infine come si possono effettuare più interlinee utilizzando solo tre istruzioni 60, 70 ed 80.

```
LIST
FILE    *PRINT2

0010 PRINT -12345678E-78;-12345678E-89;-12345678E-78;-12345678E-56;-12345678E-56
0020 PRINT -12345678912E-56,
0030 PRINT -12345678912E-23,-123456789E-62,-1234567896E-78
0040 PRINT 12
0050 PRINT " ECCO COME SI POSSONO FARE 10 INTERLINEE:"
0060 FOR I=1 TO 10 STEP 1
0070 PRINT
0080 NEXT I
0090 PRINT "SONO STATE FATTE 10 INTERLINEE!"
0100 END

END OF LISTING

RUN
**** FORMALLY CORRECT PROGRAM ****
-1.2345678E-71 -1.2345678E-82 -1.2345678E-71 -1.2345678E-49 -1.2345678E-49
-1.2345679E-46  -1.2345679E-13  -1.2345679E-54  -1.2345679E-69
 12
 ECCO COME SI POSSONO FARE 10 INTERLINEE:









SONO STATE FATTE 10 INTERLINEE!
```

3. Come si può vedere dalla esecuzione della routine sottostante, se il contenuto di una variabile di cui si comanda la stampa è superiore ad 80 caratteri, viene generato un a capo ogni 80 caratteri.

```
LIST
FILE

0010 DCL 256 A$
0020 LET A$=""
0030 FOR I=0 TO 255 STEP 1
0040 LET A$=A$+CHR$(I)
0050 NEXT I
0060 PRINT A$
0070 END

END OF LISTING


RUN
■┌┴┘└B┤o└┐>Ξ↓↑←■■■■■┤a→■↑└■■■■■ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]↑_`abcdefghijklmnopqrstuvwxyz{|}~■■■■■
```

5. In questo esempio si vede che come operàndo di una istruzione PRINT si può avere una espressione stringa (istruzione 50) od una espressione numerica (istruzione 110).

```
LIST
FILE     PRINT

0010 DCL 70 A$,40(B$,C$)
0020 LET B$="Il mese di agosto e` caldo."
0030 LET C$="Il mese di febbraio e` freddo."
0034 PRINT
0035 PRINT
0036 PRINT
0040 LET A$="La primavera e` la stagione piu` bella dell`anno!"
0050 PRINT EXT$(B$+A$+C$,28,76)
0060 LET A=10
0070 LET B=3156
0080 LET C=A*B
0090 PRINT
0100 PRINT
0110 PRINT (A*SQR(B*LGT(C))-10)/LOG(A)
0120 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****


La primavera e` la stagione piu` bella dell`anno!


 513.16547
```

Istruzione PRINT USING

Funzione

Stampa dati e testi in un formato predefinito in una istruzione immagine.

Formato

PRINT USING {**line-num** | **string-var**} , {**num-exp** | **string-exp**} [, {**num-exp** | **string-exp**}] ...

dove:

line-num
indica il numero di linea di una istruzione immagine

string-var
indica una variabile stringa il cui contenuto è la immagine con cui i valori specificati nell'istruzione devono essere stampati

num-exp
è una espressione numerica il cui valore deve essere stampato

string-exp
è una espressione stringa il cui valore deve essere stampato

Azione

I valori delle espressioni sono convertiti nel formato specificato nella istruzione immagine di formatc il cui numero di linea è indicato con line-num o nel formato specificato con il contenuto della variabile stringa string-var; sono quindi stampati da sinistra a destra nell'ordine in cui compaiono nella istruzione.

L'associazione tra i valori da stampare ed i campi della immagine di formato è fatta da sinistra a destra nell'ordine con cui i valori compaiono nell'istruzione ed i campi immagine nella immagine di formato.

Note

1. Ogni istruzione PRINT USING stampa i valori delle espressioni presenti nella istruzione a partire da una nuova riga di stampa, anche se una precedente istruzione PRINT terminava con "," o ";".

2. Se vi sono più valori da stampare, nella PRINT USING, che campi di formato, nella immagine di formato, i valori in più sono stampati sulle righe di stampa successive con lo stesso formatc.

3. Se vi sono più campi di formato, nella immagine di formato, che valori da stampare, nella PRINT USING, in corrispondenza dei campi immagine eccedenti vengono generati degli spazi.

4. I valori da stampare, nella PRINT USING, ed i campi di formato, nella immagine di formato, devono essere coerenti: ad un valore numerico deve corrispondere un campo di formato numerico etc..

5. Le costanti stringa si devono specificare racchiuse tra virgolette.

6. Per ulteriori informazioni si veda l'istruzione IMMAGINE.

7. Se la configurazione di sistema installata è priva di stampante integrata e di stampante ausiliaria (vedi comando CONFIGURE), l'istruzione PRINT visualizza sul display le informazioni relative alle espressioni in essa specificate. Ad ogni visualizzazione l'esecuzione del programma si interrompe e riprende premendo il tasto CONTINUE. Per leggere le informazioni prodotte in una linea si devono utilizzare i tasti [→], [REPEAT] e [SHIFT] come spiegato nel capitolo 1, par. "La tastiera". Se le informazioni prodotte occupano più di una linea, le informazioni relative ad ogni linea successiva alla prima sono visualizzate premendo ogni volta CONTINUE. Quando, premendo il tasto CONTINUE, il sistema emette un segnale acustico la linea visualizzata è l'ultima del testo prodotto dalla istruzione PRINT.

Esempi

1. Dalla esecuzione della seguente routine vediamo come vengono stampati i dati relativi ai campi di tipo numerico.

```
LIST
FILE    +PUSING

0010 PRINT
0020 PRINT
0030 PRINT
0040 FOR I=1 TO 4 STEP 1
0050 DCL 80A$
0060 DISP "Introduci la IMMAGINE!      ";
0070 INPUT A$
0080 DISP
0090 DISP "Introduci i dati,             ";
0100 INPUT B,C,D
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT USING A$,B,C,D
0160 PRINT
0170 PRINT
0180 PRINT
0190 PRINT
0200 NEXT I
0210 END

END OF LISTING


RUN


Introduci la IMMAGINE!      ?
Numero intero:   ##########################        **        ####

Introduci i dati,             ?
-1234567890123E11,-9,123.87



Numero intero:   -1234567890123000000000000        -9        123



Introduci la IMMAGINE!      ?
Numero decimale:  **.####################        **.#        ####.

Introduci i dati,             ?
0.123456789123E-10,0.5,123.87



Numero decimale:   0.0000000000123456789        0.5        124.



Introduci la IMMAGINE!      ?
Numero esponenziale:  ####################.#####↑↑↑↑   **.↑↑↑↑        ####.↑↑↑↑

Introduci i dati,             ?
-12345678.12345E-15,-5E-15,123.87E12
```

```
Numero esponenziale:  -123456781234500000.00000E-26   -5.E-15        124.E+12


Introduci la IMMAGINE!      ?
Numero con segno $:   $$$$$$$$$$$$$$$$$$$$$$$$$    $$$      $$$$$.

Introduci i dati.             ?
123456890123,-5,123.87


Numero con segno $:               $123456890123    $-5       $124.
```

* Per ulteriori esempi si veda l'istruzione IMMAGINE *

2. Vediamo un esempio d'impiego della funzione TAB.

```
LIST
FILE     TAB


0010 PRINT "Ecco un esempio della funzione TAB nell'istruzione PRINT"
0020 PRINT "*** Ricorda di racchiudere tra parentesi l'argomento di TAB ***"
0030 PRINT TAB(5);"A";TAB(20);"B"
0040 PRINT TAB(7);"A";TAB(22);"B"
0050 PRINT TAB(9);"A";TAB(24);"B"
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio della funzione TAB nell'istruzione PRINT
*** Ricorda di racchiudere tra parentesi l'argomento di TAB ***
    A               B
      A               B
        A               B
```

Istruzione RANDOMIZE

Funzione

Permette la generazione di numeri casuali.

Formato

RAN [DOMIZE]

Azione

L'istruzione RANDOMIZE, posta prima di una istruzione che richiama la funzione di sistema RND, fa in modo che i valori numerici ritornati da RND siano casuali non solo nell'ambito della esecuzione del programma, ma anche tra diverse esecuzioni dello stesso programma.

Nota

Quando si reinizializza il sistema e si riesegue il programma è rigenerata la stessa sequenza di numeri casuali.

Esempio

Vediamo nella stampa sottostante come la prima routine, ogni volta che è eseguita, produce la stessa sequenza di numeri casuali compresi tra zero ed uno. Se prima della routine suddetta, si pone l'istruzione RANDOMIZE (si noti che l'istruzione può essere introdotta abbreviando la parola RANDOMIZE in RAN), ogni volta che la nuova routine è eseguita si ottiene una sequenza diversa di numeri casuali compresi tra zero e uno.

```
LIST
FILE    +RAN

0010 FOR I=1 TO 20 STEP 1
0030 PRINT RND,
0040 NEXT I
0050 END

END OF LISTING


RUN
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105
 4.7693357E-02    .68675523        .17611750        3.5848356E-02    .50296996
 .62662024        .88834013        .85254312        .73719855        .70058045
RUN
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105
 4.7693357E-02    .68675523        .17611750        3.5848356E-02    .50296996
 .62662024        .88834013        .85254312        .73719855        .70058045
5 RAN
LIST
FILE    +RAN

0005 RAND
0010 FOR I=1 TO 20 STEP 1
0030 PRINT RND,
0040 NEXT I
0050 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 .46928183        .93340150        .73981330        .93170842        .66550775
 4.9468396E-02    .41167724        .16474361        4.9257438E-02    .45449761
 .29001342        .73709782        1.0500667E-02    .92415374        .38637937
 .26806888        .15637405        7.7169896E-03    .18172449        3.9252707E-03
RUN
 .30027045        .84385646        .14483391        .59239684        .91177673
 .51377738        .21853162        .35957285        .80906657        .55912861
 .96844639        .33748557        .87592708        .94394928        .14757729
 .35964649        .44423446        .63348580        .61190501        .13459401
```

Istruzione READ

Funzione

Assegna alle variabili di programma specificate i valori contenuti nel file dati, interno al programma, generato con le istruzioni DATA.

Formato

READ { **num-var** | **string-var** } [, { **num-var** | **string-var** }] ...

dove:

num-var
è una variabile numerica, semplice o con indice, alla quale viene assegnato un valore numerico prelevato dal file dati interno generato con le istruzioni DATA del programma

string-var
è una variabile stringa, semplice o con indice, alla quale viene assegnato un valore numerico prelevato dal file dati interno suddetto

Azione

I valori nel file dati interno sono assegnati nell'ordine (da sinistra a destra) alle variabili indicate nella istruzione READ, iniziando dalla posizione indicata in quel momento dal pointer del file dati interno (vedi istruzione DATA).

Note

1. Il pointer del file dati interno può essere riposizionato all'inizio del file stesso mediante l'istruzione RESTORE (vedi istruzione RESTORE).

2. I valori degli indici delle variabili con indice sono assegnati nel momento in cui compaiono nelle istruzioni READ; così una variabile presente in una istruzione READ può essere utilizzata come indice in una successiva variabile con indice presente nella stessa istruzione READ.

3. I valori assegnati dal file dati interno devono essere coerenti con il tipo di variabili presenti nella istruzione READ.

4. Se c'è una istruzione READ in un programma deve esserci almeno una istruzione DATA.

5. Se vi è una variabile in una istruzione READ a cui non può essere assegnato un valore perchè il file dati interno è esaurito, allora l'esecuzione del programma è sospesa; premendo BREAK il sistema commuta nello stato comandi.

6. Per ulteriori informazioni si veda l'istruzione DATA.

Esempio

1. Nella routine seguente si vede che è possibile, con una sola istruzione READ, assegnare il valore all'indice di una variabile con indice e quindi assegnare un valore alla variabile stessa.

```
LIST
FILE    +READ

0005 LET I=9
0006 LET A(9)=45
0010 DATA 1,2
0020 READ I,A(I)
0030 PRINT "I=";I,"A(1)=";A(1),"A(9)=";A(9)
0040 END

END OF LISTING


RUN
I= 1            A(1)= 2         A(9)= 45
```

Istruzione READ:

Funzione

Assegna alle variabili di programma specificate i valori contenuti in un file esterno.

Formato

READ: file-designator, {**num-var** | **string-var**} [, {**num-var** | **string-var**}] ... [**EOF line-num**]

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica da quale file dati esterno devono essere prelevati i dati da assegnare alle variabili specificate nella istruzione

num-var
è una variabile numerica, semplice o con indice; alla quale è assegnato il relativo dato numerico prelevato dal file esterno, specificato con file-designator

string-var
è una variabile stringa, semplice o con indice, alla quale è assegnato il dato prelevato dal file dati esterno specificato con file-designator

line-num
è il numero di linea della istruzione a cui viene ceduto il controllo della esecuzione del programma se è raggiunta la fine logica (file sequenziale) o la fine fisica (file ad accesso diretto) del file dati specificato con file-designator

Azione

L'espressione numerica è eseguita ed il valore ottenuto, arrotondato all'intero più prossimo nd, costituisce il numero designatore del file esterno da cui sono prelevati i valori dei dati da assegnare alle variabili di programma indicate nella istruzione.

I valori dei dati sono prelevati dal file con numero

designatore nd iniziando dal dato su cui è posizionato il pointer del file ed assegnati nell'ordine (da sinistra a destra) alle variabili presenti nella istruzione.

Il valore del pointer del file esterno è aggiornato in modo da indirizzare la posizione successiva all'ultimo dato letto.

Se il pointer di un file dati sequenziale indirizza una posizione successiva all'ultimo dato registrato nel file, l'esecuzione dell'istruzione READ: dà una segnalazione d'errore e l'esecuzione del programma è sospesa, se, però, è presente l'opzione EOF, il controllo della esecuzione del programma passa all'istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Se il pointer di un file dati ad accesso diretto indirizza la posizione successiva all'ultima parola allocata per il file (vedi comando CREATE), l'esecuzione dell'istruzione READ: dà una segnalazione di errore e l'esecuzione del programma è sospesa; se è presente l'opzione EOF il controllo della esecuzione del progamma passa alla istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Note

1. Con l'istruzione READ: si può leggere anche il contenuto di un file testo. Il file testo è considerato un file dati sequenziale contenente stringhe di caratteri ognuna composta da una linea del file testo con incluso il numero di linea.

2. Per poter eseguire una istruzione READ: il file dati esterno deve essere stato prima aperto all'accesso da parte del programma mediante una istruzione FILES o FILE:.

3. Dopo l'esecuzione di una istruzione WRITE:, se il file dati è di tipo sequenziale, si deve eseguire una istruzione RESTORE: (vedi istruzione RESTORE:) prima di una istruzione READ:.

4. Se il file è stato dichiarato, con il comando CREATE (vedi capitolo 3) ad <u>accesso diretto</u> si può leggere qualunque dato utilizzando prima dell'i-

struzione READ: l'istruzione SETW: (vedi istruzione SETW:) per assegnare al pointer l'indirizzo del dato che si vuol leggere.

5. I dati letti dal file devono essere coerenti con le variabili specificate nell'istruzione READ:.

6. Il risultato della espressione numerica (specificata con file-designator), arrotondato all'intero più prossimo, deve essere maggiore di Ø e minore o uguale al numero di file che possono essere utilizzati contemporaneamente dal programma (vedi istruzione FILES).

7. In una istruzione READ: si può specificare una variabile utilizzata come indice in una variabile con indice specificata successivamente nella stessa istruzione.

8. Se ad una variabile stringa viene assegnata, da un file dati esterno, una stringa con più caratteri di quelli definiti per la sua lunghezza di allocazione, la stringa è assegnata alla variabile suddetta troncata dei caratteri eccedenti sulla destra. Il sistema è nello stato di debugging e visualizza un messaggio di errore di tipo recuperabile. Premendo il tasto di console CONTINUE l'esecuzione del programma continua; premendo BREAK l'esecuzione termina.

9. Se viene assegnato ad una variabile numerica, rappresenta in singola precisione, un valore numerico, rappresentato su file dati in doppia precisione, con esponente al di fuori del range della singola precisione, il sistema le assegna il valore zero, se l'esponente è nella zona di UNDERFLOW per la singola precisione, od il valore +9.99999E63 (oppure -9.99999E63), se l'esponente è nella zona di OVERFLOW per la singola precisione. Il sistema visualizza un messaggio di errore recuperabile ed è nello stato di debugging. Premendo il tasto di console CONTINUE l'esecuzione continua; premendo BREAK l'esecuzione del programma termina.

Esempi

1. Nella routine sottostante si mostra che se il numero designatore è espresso con un valore che arrotondato all'intero più prossimo è minore di 1 o maggiore del numero di file accessibile contemporaneamente da programma (ricavato dall'istruzione FILES), è segnalato un errore di tipo non recuperabile (vedi sotto le prime due esecuzioni dopo la digitazione dei primi due comandi RUN) e l'esecuzione del programma è sospesa. Premendo BREAK il sistema commuta nello stato comandi. Si notino gli altri casi: ad esempio quando il numero introdotto è 1.8, il file letto è quello con numero designatore 2, etc.

```
LIST
FILE


0010 FILES APPEND;ISOT;DIRET1;C1
0020 FOR I=1 TO 6 STEP 1
0030 DISP "Quale file vuoi leggere";
0040 INPUT A
0050 READ :A,B
0052 IF A-INT(A)<=0.5 THEN 57
0053 LET A=INT(A)+1
0054 GOTO 60
0057 LET A=INT(A)
0060 PRINT "Ho prelevato";B;"nel file indicato nella posizione";A;"in FILES."
0070 NEXT I
0080 END

END OF LISTING


RUN
Quale file vuoi leggere?
0
ERROR 78  IN LINE 50
RUN
Quale file vuoi leggere?
5
ERROR 77  IN LINE 50
RUN
Quale file vuoi leggere?
1.3
Ho prelevato 1 nel file indicato nella posizione 1 in FILES.
Quale file vuoi leggere?
1.8
Ho prelevato 0 nel file indicato nella posizione 2 in FILES.
Quale file vuoi leggere?
3.2
Ho prelevato 1 nel file indicato nella posizione 3 in FILES.
Quale file vuoi leggere?
3.75
Ho prelevato 1 nel file indicato nella posizione 4 in FILES.
Quale file vuoi leggere?
4.4
Ho prelevato 8 nel file indicato nella posizione 4 in FILES.
Quale file vuoi leggere?
2
Ho prelevato 0 nel file indicato nella posizione 2 in FILES.
```

2. Nella routine sottostante si può vedere come nella istruzione READ: si può assegnare un valore da un file dati esterno ad una variabile che è utilizzata nella stessa istruzione come indice di una variabile con indice.

```
LIST
FILE


0010 FILES DIRET1
0020 LET I=A(10)=10
0030 READ :1,I,A(I)
0040 PRINT "I=";I,"A(1)=";A(1),"A(10)=";A(10)
0050 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
I= 1            A(1)= 2         A(10)= 10
```

3. Vediamo un esempio d'impiego dell'opzione EOF in un'istruzione READ: che legge i dati di un file sequenziale.

```
LIST
FILE


0010 DCL 80A$
0020 FILES Q
0030 FOR I=1 TO 5 STEP 1
0040 READ :1,A$ EOF 70
0050 PRINT A$
0060 NEXT I
0070 PRINT "Nel file Q non vi sono piu` dati!"
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
1234567890123456789
Nel file Q non vi sono piu` dati!
```

4. Con il programma sottostante sono registrati nel file ad accesso diretto PROVA, creato in una libreria allocando per esso 128 byte, 30 dati numerici in singola precisione, per cui rimangono le due ultime parole ancora non registrate (ad esse il sistema aveva assegnato, in fase di inizializzazione, il valore zero). Quando il file PROVA viene letto dall'inizio ad un certo punto il sistema si pone nello stato di debugging e visualizza il messaggio recuperabile ERROR 1; premendo il tasto CONTINUE la situazione si ripete ma ad una successiva pressione di CONTINUE l'esecuzione continua fino al termine del file. Solo quando è raggiunta la fine fisica del file PROVA entra in azione l'opzione EOF che rimanda la esecuzione alla istruzione 130.

```
0010 FILES PROVA
0020 DCL SA
0030 LET A=0
0040 FOR I=1 TO 30 STEP 1
0050 LET A=I
0060 WRITE :1,A
0070 NEXT I
0080 SETW :1 TO 1
0085 FOR I=1 TO 40 STEP 1
0090 READ :1,B EOF 130
0100 PRINT B,
0120 NEXT I
0130 PRINT "Sono giunto all'ultima parola del file PROVA!"
0140 END

END OF LISTING


RUN
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
ERROR 1   IN LINE 90
ERROR 1   IN LINE 90
 0              0              Sono giunto all'ultima parola del file PROVA!
```

5. Vediamo qui una routine che legge un file testo registrato in una libreria col nome MANUAL. Sono lette e stampate le prime 20 linee del file testo.

```
FILE

0010 DCL 80(A$)
0020 FILES MANUAL
0030 FOR I=1 TO 20 STEP 1
0040 READ :1,A$
0050 PRINT A$
0060 NEXT I
0070 END

END OF LISTING


0010
0020
0030
0040                        4. BASIC: DATI, VARIABILI, ESPRESSIONI E FILE DATI
0050
0060  Questo capitolo offre al lettore che ha gia'una certa familiarita' con i
0070 concetti del linguaggio BASIC la possibilita' di effettuare una rapida e
0080 completa consultazione di tutte le possibilita' offerte dal linguaggio
0090 BASIC realizzato per il sistema P6066.
0100
0110
0120   4.1 CARATTERI DEL LINGUAGGIO BASIC
0130
0140  I caratteri che hanno un ruolo sintattico nel linguaggio BASIC sono
0150 classificabili in tre categorie:
0160
0170                                  CARATTERI ALFABETICI
0180                                  CARATTERI NUMERICI
0190                                  CARATTERI SPECIALI
0200
```

Istruzione REMARK

Funzione

Permette di inserire in un programma dei commenti che rendono facile la lettura del relativo listing.

Formato

REM [ARK] comment

dove:
comment
è una successione di caratteri del set ISO (vedi appendice E) che rappresenta un commento per il programmatore.

Azione

Non è una istruzione eseguibile. Il commento appare nel listing del programma, ma non produce alcun effetto durante l'esecuzione del programma.

Esempio

Come si vede dall'esempio l'istruzione REMARK può essere introdotta abbreviando la parola chiave in REM.

```
NEW
10 REM Questa istruzione ti dice che cosa fanno le istruzioni che seguono.
20 END
LIST
FILE


0010 REM Questa istruzione ti dice che cosa fanno le istruzioni che seguono.
0020 END

END OF LISTING
```

Istruzione RESTORE

Funzione

Posiziona il pointer del file dati interno all'inizio del file stesso.

Formato

RESTORE

Azione

Il pointer del file dati interno (vedi istruzione DATA) viene rimosso dalla posizione attuale e posizionato all'inizio del file stesso.

Note

1. Una istruzione RESTORE dopo una precedente istruzione RESTORE (senza istruzioni READ interposte tra le due istruzioni suddette) non provoca alcun effetto.

2. Una istruzione RESTORE in un programma senza istruzioni DATA non provoca alcun effetto.

Esempio

La routine sottostante mostra un impiego dell'istruzione RESTORE.

```
LIST
FILE     RESTOR

0010 DATA 1,2,4,5,6,7,8,9
0020 FOR I=1 TO 7 STEP 1
0030 READ I
0040 PRINT I
0050 NEXT I
0060 RESTORE
0070 READ A,B,C
0080 PRINT
0090 PRINT A,B,C
0100 END

END OF LISTING
```

RUN
1
2
4
5
6
7

1 2 4

Istruzione RESTORE:

Funzione

Posiziona il pointer di un file esterno all'inizio del file e, se il file è di tipo sequenziale, ne permette la lettura.

Formato

RESTORE: file-designator

dove:
file-designator
è una espressione numerica il cui valore arrotondato all'intero più prossimo indica un designatore di file (vedi istruzioni FILES e FILE:).

Azione

L'espressione numerica viene eseguita ed il valore calcolato è arrotondato all'intero più prossimo nd.

Il puntatore del file esterno il cui numero designatore è nd è posizionato all'inizio del file stesso.

Note

1. Se il file è sequenziale, dopo l'esecuzione dell'istruzione RESTORE:, può essere letto con l'istruzione READ:

2. Se il file con numero designatore nd è di tipo ad accesso diretto, l'esecuzione dell'istruzione RESTORE: posiziona il pointer all'inizio del file.

3. Il valore della istruzione numerica arrotondato all'intero più prossimo deve essere maggiore di zero e minore od uguale al numero di file contemporaneamente aperti all'accesso da parte del programma, stabilito con l'istruzione FILES.

4. L'istruzione RESTORE: può essere usata con riferimento ad un file testo e pone il pointer all'inizio del file medesimo.

Esempi

1. Vediamo un impiego dell'istruzione RESTORE: con un file di tipo sequenziale.

```
LIST
FILE

0010 FILES FILES1
0020 FOR I=1 TO 16 STEP 1
0030 READ :1,A
0040 PRINT A
0050 NEXT I
0060 RESTORE :1
0070 FOR I=1 TO 6 STEP 1
0080 READ :1,B
0090 PRINT "B=";B
0100 NEXT I
0110 RESTORE :1
0120 READ :1,C
0130 PRINT "C=";C
0140 END

END OF LISTING


RUN
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
B= 1
B= 2
B= 3
B= 4
B= 5
B= 6
C= 1
```

2. Con questa routine si rileggono i dati di un file ad accesso diretto dall'inizio. Si noti come l'istruzione SETW: 1 TO 1 svolge l'azione svolta nella routine precedente per il file sequenziale dalla istruzione RESTORE:.

```
LIST
FILE    *RESTO3

0010 FILES FILED2
0020 FOR I=1 TO 10 STEP 1
0030 READ :1,A
0040 PRINT "A=";A
0050 NEXT I
0060 SETW :1 TO 1
0070 READ :1,B,C,D
0080 PRINT "B=";B,"C=";C,"D=";D
0090 END

END OF LISTING


RUN
A= 1
A= 2
A= 3
A= 4
A= 5
A= 6
A= 7
A= 8
A= 9
A= 10
B= 1            C= 2            D= 3
```

Istruzione RETURN

Funzione

Trasferisce il controllo della esecuzione di un programma all'istruzione successiva ad una istruzione GOSUB oppure ON...GOSUB.

Formato

RETURN

Azione

Passa il controllo della esecuzione del programma alla prima istruzione esecutiva successiva alla istruzione GO SUB, o ON...GOSUB, che ha ceduto il controllo della esecuzione del programma al sottoprogramma di cui l'istruzione RETURN fa parte.

Note

1. In un sottoprogramma vi possono essere più istruzioni RETURN.

2. Per una facile lettura del programma, è bene utilizzare una sola istruzione RETURN in un sottoprogramma alla quale ci si può riferire da diversi punti del sottoprogramma stesso mediante delle istruzioni GOTO o IF...THEN.

Esempi

Vedi istruzioni GOSUB ed ON...GOSUB.

Istruzione RKB

Funzione

Assegna ad una variabile stringa i caratteri introdotti da tastiera che sono scelti tra i caratteri del set ISO (vedi appendice E).

Formato

RKB string-var

dove:
string-var
è una variabile stringa, semplice o con indice, a cui vengono assegnati i caratteri introdotti da tastiera.

Azione

L'esecuzione del programma è interrotta e sul display è visualizzato il carattere ?.

La stringa di caratteri digitata è assegnata alla variabile indicata nella istruzione.

Note

1. L'istruzione RKB permette di assegnare ad una variabile di programma il carattere virgolette (").

2. Se i caratteri introdotti superano la lunghezza di allocazione della variabile stringa specificata, il sistema assegna alla variabile la stringa troncata dei caratteri eccedenti la lunghezza di allocazione e commuta nello stato di debugging visualizzando un un messaggio sul display.

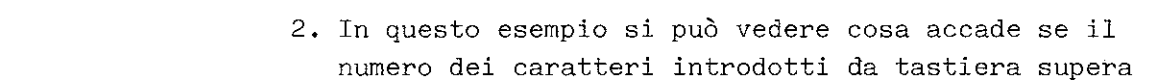
Esempi

1. Vediamo in questo esempio come, utilizzando RKB, si possono assegnare ad una variabile stringa i caratteri che si vogliono. In particolare si noti che si possono introdurre frasi comprese tra virgolette; infatti anche il carattere virgolette è considerato

parte del valore da assegnare alla variabile specificata. Si noti come la variabile stringa può essere una variabile con indice.

```
0010 DCL 80A$,80A$()
0020 RKB A$
0030 RKB A$(1)
0040 PRINT
0050 PRINT "A$=";A$
0060 PRINT
0070 PRINT "A$(1)=";A$(1)
0080 END
RUN
**** FORMALLY CORRECT PROGRAM ****
?
"Come si vede si possono assegnare i caratteri che si vogliono!"
?
```

```
A$="Come si vede si possono assegnare i caratteri che si vogliono!"
```



2. In questo esempio si può vedere cosa accade se il numero dei caratteri introdotti da tastiera supera la lunghezza di allocazione della variabile specificata nell'istruzione RKB. Il sistema commuta nello stato di debugging e visualizza il messaggio di errore sottoriportato. Premendo CONTINUE l'esecuzione del programma riprende ed alla variabile sono assegnati i primi 16 caratteri digitati (perchè la lunghezza di allocazione di A$ è di 16 caratteri).

```
NEW
10 DISP"INTRODUCI I CARATTERI CHE VUOI    ";
20 RKB A$
30 PRINT "A$=";A$
40 END
RUN
**** FORMALLY CORRECT PROGRAM ****
INTRODUCI I CARATTERI CHE VUOI    ?
1234567890123456789
ERROR 8   IN LINE 20
A$=1234567890123456
```

Istruzione SCRATCH:

Funzione

Posiziona il pointer di un file dati esterno, di tipo sequenziale, all'inizio del file e permette di registrare in esso dei dati con una successiva istruzione WRITE:.

Formato

SCRATCH: file-designator

dove:
file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di file (vedi istruzioni FILES e FILE:).

Azione

L'espressione numerica è eseguita ed il valore ottenuto è arrotondato all'intero più prossimo nd.

Il pointer del file esterno il cui numero designatore è nd è posizionato all'inizio del file ed il contenuto del file è cancellato. Il file è posto in modalità di scrittura, ossia si possono eseguire delle istruzioni WRITE: per registrare in esso dei dati.

Note

1. Il valore dell'espressione numerica, arrotondato all'intero più prossimo, deve essere maggiore di zero e minore od uguale al numero di file dichiarati accessibili contemporaneamente da programma mediante l'istruzione FILES.

2. La successiva istruzione di I/O deve essere una istruzione WRITE: e non READ:.

3. L'istruzione SCRATCH: può essere usata solo con file sequenziali.

Esempio

Ecco un esempio di impiego dell'istruzione SCRATCH:. Si noti come l'istruzione SCRATCH: cancella il precedente contenuto del file FILES1; infatti dopo aver letto il quinto dato l'istruzione READ: esegue l'opzione EOF che indica la mancanza di ulteriori dati nel file.

```
OLD *SCRATC
LIST
FILE    *SCRATC


0010 FILES FILES1
0020 FOR I=1 TO 10 STEP 1
0030 READ :1,A
0040 PRINT "A=";A,
0050 NEXT I
0060 SCRATCH :1
0070 WRITE :1,21,22,23,24,25
0080 RESTORE :1
0090 FOR I=1 TO 10 STEP 1
0100 READ :1,B EOF 130
0110 PRINT "B=";B,
0120 NEXT I
0130 PRINT "Non si possono leggere altri dati nel file FILES1 perche`... "
0140 PRINT "... l'istruzione SCRATCH li ha cancellati."
0150 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A= 1            A= 2            A= 3            A= 4            A= 5
A= 6            A= 7            A= 8            A= 9            A= 10
B= 21           B= 22           B= 23           B= 24           B= 25
Non si possono leggere altri dati nel file FILES1 perche`...
... l'istruzione SCRATCH li ha cancellati.
```

Istruzione SETW:

Funzione

Posiziona il pointer all'inizio della parola specificata, di un file dati esterno ad accesso diretto.

Formato

SETW: file-designator TO word-num

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di file (vedi istruzioni FILES e FILE:)

word-num
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di parola nell'ambito del file dati specificato con file-designator.

Azione

Le espressioni numeriche sono eseguite e i valori ottenuti sono arrotondati agli interi più prossimi: nd (designatore di file) e np (designatore di parola).

Il puntatore del file di numero designatore nd è posizionato all'inizio della np-esima parola del file stesso.

Note

1. Il file di numero designatore nd deve essere ad accesso diretto.

2. Dopo l'istruzione SETW: si possono registrare dati sul file (istruzione WRITE:) o leggere dati dal file (istruzione READ:).

3. Il risultato della espressione numerica, arrotondato all'intero più prossimo nd, deve essere maggiore di zero e minore od uguale al numero di file che

sono contemporaneamente accessibili dal programma, come specificato dall'istruzione FILES.

4. Il risultato della espressione numerica, arrotondato all'intero prossimo np, deve essere maggiore di zero e minore od uguale al numero di parole allocate per il file con il comando CREATE (vedi comando CREATE).

Esempio

Col seguente programma si registrano nei file dati ad accesso diretto FILED1, FILED2, FILED3 e FILED4 i numeri 1,2,3 e 4 per tutta l'estensione dei file. Si noti che i numeri sono assegnati a variabili dichiarate in singola precisione per cui sono registrati ognuno su di una parola dei file suddetti. Quindi si moltiplicano i dati registrati rispettivamente nelle parole 5, 16, e 8 dei file FILED2, FILED3 e FILED4. Il risultato è registrato nella parola 3 del file FILED1. Per poter analizzare il risultato prodotto dal programma nei quattro file suddetti si vedano le stampe riportate che si riferiscono al contenuto dei file. La prima stampa è relativa al contenuto del file FILED1, la seconda al file FILED2 etc..

```
CIS
FILE    *SETW2

0004 DCL SINGLE
0005 FILES FILED1;FILED2;FILED3;FILED4
0010 FOR J=1 TO 4 STEP 1
0020 LET A=J
0030 GOSUB 200
0040 NEXT J
0050 SETW :2 TO 5
0060 WRITE :2,10
0070 SETW :3 TO 16
0080 WRITE :3,15
0100 SETW :4 TO 8
0110 WRITE :4,8
0120 REM Ora moltiplico i dati conenuti nelle parole 5, 16 e 8 dei file  ...
0130 REM ... FILED2,FILED3 e FILED4 e metto il risultato nella parola 3 ...
0140 REM ... del file FILED1.
0145 SETW :2 TO 5
0146 SETW :3 TO 16
0147 SETW :4 TO 8
0150 READ :2,X
0151 READ :3,Y
0152 READ :4,Z
0160 SETW :1 TO 3
0170 WRITE :1,X*Y*Z
0174 SETW :1 TO 3
0175 READ :1,C
0180 PRINT "Questo e`il numero posto in FILED1:";C
0190 GOTO 250
0200 SETW :A TO 1
```

```
0210 FOR I=1 TO 32 STEP 1
0220 WRITE :A,A
0230 NEXT I
0240 RETURN
0250 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****

Questo e'il numero posto in FILED1: 1200

                FILED1
1               1               1200            1               1
1               1               1               1               1
1               1               1               1               1
1               1               1               1               1
1               1               1               1               1
1               1               1               1               1
1               1               1               1               1

                FILED2
2               2               2               2               10
2               2               2               2               2
2               2               2               2               2
2               2               2               2               2
2               2               2               2               2
2               2               2               2               2
2               2               2               2               2

                FILED3
3               3               3               3               3
3               3               3               3               3
3               3               3               3               3
15              3               3               3               3
3               3               3               3               3
3               3               3               3               3
3               3               3               3               3

                FILED4
4               4               4               4               4
4               4               8               4               4
4               4               4               4               4
4               4               4               4               4
4               4               4               4               4
4               4               4               4               4
4               4               4               4               4
```

Istruzione STOP

Funzione

Interrompe l'esecuzione di un programma e commuta il sistema nello stato di debugging.

Formato

STOP

Azione

L'esecuzione del programma è sospesa.

Sul display è visualizzato il messaggio "STOP numero di linea" dove "numero di linea" è quello della istruzione stessa.

Le variabili di programma conservano i loro contenuti ed il sistema si trova nello stato di debugging, per cui si possono effettuare tutte le operazioni descritte nel capitolo 7.

Note

1. L'istruzione STOP è utile nella fase di debugging (vedi capitolo 7) del programma.

2. L'istruzione STOP non deve essere compresa in una funzione multilinea.

3. L'esecuzione del programma riprende non appena si preme il tasto CONTINUE od il tasto STEP. Per maggiori dettagli sull'impiego dei due tasti suddetti si veda il capitolo 7.

4. L'istruzione STOP è anche utile per fermare l'esecuzione del programma e permette all'operatore di digitare uno dei tasti funzione che può rinviare l'esecuzione ad una routine scelta dall'operatore stesso (si veda l'esempio allegato all'istruzione FKEY#).

5. Se l'istruzione STOP è preceduta da una istruzione DISP con alla fine il punto e virgola, o la virgola, allora il messaggio relativo all'istruzione DISP permane sul display ed il messaggio "STOP numero di linea" non viene visualizzato.

Esempi

Si vedano gli esempi indicati nei capitoli 7, 8 e nella istruzione FKEY #.

Istruzione TRACE OFF

Funzione	Termina la stampa dei numeri di linea delle istruzioni di programma eseguite.
Formato	**TRACE OFF**
Azione	La stampa dei numeri di linea delle istruzioni esecutive del programma, predisposta con la precedente istruzione TRACE ON, è interrotta. La luce di console TRACE si spegne.
Note	1. Se l'istruzione TRACE OFF non esiste nel programma, o comunque non è incontrata durante l'esecuzione del programma, la stampa dei numeri di linea continua fino alla esecuzione della istruzione END. 2. L'istruzione TRACE OFF annulla anche l'effetto prodotto premendo il tasto di console TRACE ON. Se dopo si preme di nuovo il tasto di console TRACE il sistema riprende a stampare i numeri di linea delle istruzioni esecutive.
Esempi	Si veda l'esempio 2 dell'istruzione TRACE ON.

Istruzione TRACE ON

Funzione

Esegue la stampa del numero di linea di ogni successiva istruzione di programma eseguita.

Formato

TRACE ON

Azione

I numeri di linea delle istruzioni esecutive successive vengono stampati nell'ordine in cui sono eseguite. La luce del tasto di console TRACE si accende.

Note

1. Non sono stampati i numeri di linea delle istruzioni non esecutive.

2. I numeri di linea delle istruzioni delle funzioni definite dall'utente (monolinea e multilinea) non sono stampati.

3. Il numero di linea della istruzione TRACE ON non viene stampato, mentre viene stampato il numero di linea della istruzione TRACE OFF.

4. I numeri di linea stampati sono interposti tra altre linee di stampa comandate da programma.

5. L'effetto prodotto da TRACE ON viene annullato quando è eseguita l'istruzione TRACE OFF; se non vi è una istruzione TRACE OFF allora la stampa dei numeri di linea continua fino all'istruzione END (anche di essa è stampato il numero di linea).

6. L'istruzione TRACE ON è utile durante il debugging di un programma (si veda il capitolo 7).

7. Se si preme il tasto di console TRACE si ottiene lo stesso effetto prodotto con l'esecuzione della istruzione TRACE ON. In questo caso anche il numero

di linea della istruzione TRACE ON è stampato. Quindi, se prima di introdurre il comando RUN si preme il tasto TRACE i numeri di linea delle istruzioni esecutive sono stampati dall'inizio anche se l'istruzione TRACE ON viene eseguita più avanti nel programma.

Esempi

1. Nel programma seguente l'istruzione TRACE ON è la prima e non vi è una istruzione TRACE OFF, per cui tutti i numeri di linea delle istruzioni esecutive vengono stampati. Si noti come i numeri di linea sono preceduti dal segno # e non sono stampati sulla stessa linea di altre stampe prodotte dal programma.

```
LIST
FILE    *TRAC1

0005 TRACE ON
0010 REM Ecco un esempio di impiego dell'istruzione TRACEON
0020 DCL SA
0030 DATA 1,2,3,4,5
0040 READ A,B,C,D,E
0050 FILES FILED1;FILED2;FILED3
0060 SETW :1 TO 10
0065 READ :1,A
0070 PRINT "Ho letto questo dato nel file FILED1:";A
0080 SETW :2 TO 4
0090 READ :2,A
0100 PRINT "Ho letto questo dato nel file FILED2:";A
0110 SETW :3 TO 8
0120 READ :3,A
0130 PRINT "Ho letto questo dato nel file FILED3"
0150 LET X=B*C*D*E
0160 PRINT "X=";X
0170 END

END OF LISTING


RUN
#40
#60
#65
#70
Ho letto questo dato nel file FILED1: 1
#80
#90
#100
Ho letto questo dato nel file FILED2: 2
#110
#120
#130
Ho letto questo dato nel file FILED3
#150
#160
X= 120
#170
```

2. Il programma sottostante è identico al precedente con l'aggiunta dell'istruzione TRACE OFF che, come si vede, annulla l'effetto prodotto dalla istruzione TRACE ON. Si noti come sia stampato anche il numero di linea dell'istruzione TRACE OFF (95).

```
LIST
FILE    *TRAC1

0005 TRACE ON
0010 REM Ecco un esempio di impiego dell'istruzione TRACEON
0020 DCL SA
0030 DATA 1,2,3,4,5
0040 READ A,B,C,D,E
0050 FILES FILED1;FILED2;FILED3
0060 SETW :1 TO 10
0065 READ :1,A
0070 PRINT "Ho letto questo dato nel file FILED1:";A
0080 SETW :2 TO 4
0090 READ :2,A
0095 TRACE OFF
0100 PRINT "Ho letto questo dato nel file FILED2:";A
0110 SETW :3 TO 8
0120 READ :3,A
0130 PRINT "Ho letto questo dato nel file FILED3"
0150 LET X=B*C*D*E
0160 PRINT "X=";X
0170 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
#40
#60
#65
#70
Ho letto questo dato nel file FILED1: 1
#80
#90
#95
Ho letto questo dato nel file FILED2: 2
Ho letto questo dato nel file FILED3
X= 120
```

Istruzione WHERE:

Funzione

Permette di determinare su quale parola di un file dati esterno è posizionato il relativo pointer ed il tipo di dato da esso indirizzato.

Formato

WHERE: file-designator, num- var$_1$ [, num-var$_2$ [, num-var$_3$]]

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica il designatore di un file dati

num-var$_1$
è una variabile numerica alla quale viene assegnato il valore corrente (in numero di parole) del pointer del file esterno specificato con file-designator

num-var$_2$
è una variabile numerica alla quale è assegnato un valore numerico che specifica il dato su cui è posizionato il pointer

num-var$_3$
è una variabile numerica che, se il tipo di dato su cui è posizionato il pointer è una stringa di caratteri, ne specifica la lunghezza.

Azione

Il valore attuale del pointer del file dati esterno specificato con il valore arrotondato all'intero più prossimo dell'espressione numerica file-designator è assegnato alla variabile numerica num-var$_1$.

Alla variabile num-var$_2$ è assegnato uno dei seguenti valori numerici:

Valore	Interpretazione
0	Se nella posizione corrente del pointer non è riconosciuto alcun identificatore
1	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico rappresentato in singola precisione (pointer posizionato all'inizio di un dato numerico in singola precisione)
2	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico rappresentato in doppia precisione (pointer posizionato all'inizio di un dato numerico in doppia precisione)
3	se nella posizione corrente del pointer è riconosciuto un identificatore di un dato stringa (pointer posizionato all'inizio di una stringa di caratteri)
4	se il pointer è posizionato alla fine del file dati esterno
5	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico non inizializzato rappresentato in singola precisione
6	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico non inizializzato rappresentato in doppia precisione
7	se nella posizione corrente del pointer è riconosciuto un identificatore di dato stringa non inizializzato

Alla variabile $\underline{\text{num-var}}_3$ è assegnato il numero di caratteri che compongono la stringa su cui è posizionato il pointer, quando il valore di $\underline{\text{num-var}}_2$ è 3; se il valore di $\underline{\text{num-var}}_2$ è diverso da 3, la variabile $\underline{\text{num-var}}_3$ è posta a zero.

Note

1. Si osservi che anche se il pointer è posizionato all'interno di una stringa di caratteri, particolari codici vengono riconosciuti come identificatore di inizio dato.

2. Gli operandi $\underline{\text{num-var}}_2$ e $\underline{\text{num-var}}_3$ si possono specificare se il file dati esterno è stato aperto in lettura.

Esempio

Nel programma seguente l'istruzione WHERE: permette di leggere e stampare solamente le stringhe di dati che sono presenti nel file DATI. La funzione multilinea FNA fa in modo che l'esecuzione del programma non sia interrotta quando è raggiunta la fine del file.

```
LIST
FILE     WHERE

0010 LET C1=0
0020 INTERRUPT ENABLE (I,"A")
0030 FILES DATI
0040 WHERE : 1,P,T
0050 IF C1=1 THEN 240
0060 IF T=3 THEN 100
0070 IF T=1 THEN 160
0080 IF T=2 THEN 140
0090 SETW :1 TO P+1
0100 READ :1,A$
0110 PRINT "LA PAROLA ";P;"CONTIENE: ";A$
0130 GOTO 40
0140 SETW :1 TO P+2
0150 GOTO 40
0160 SETW :1 TO P+1
0180 GOTO 40
0190 DEF FNA(C,L)
0200 LET C1=1
0210 LET P=0
0220 LET FN*=0
0230 FNEND
0240 END

END OF LISTING


LA PAROLA  1 CONTIENE: OLIVETTI
LA PAROLA  4 CONTIENE: P6066
LA PAROLA  8 CONTIENE: NEW YORK
LA PAROLA  11 CONTIENE: ROMA
LA PAROLA  60 CONTIENE: MILANO
LA PAROLA  100 CONTIENE: GINEVRA
```

Istruzione WRITE:

Funzione

Registra in un file dati esterno i valori delle espressioni specificate.

Formato

WRITE: file-designator, {num-exp | string-exp} [, {num-exp | string-exp}] ... [EOF line-num]

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo nd, indica il designatore di un file (vedi istruzioni FILES e FILE:)

num-exp
è una espressione numerica il cui valore è registrato sul file dati esterno specificato con file-designator

string-exp
è una espressione stringa il cui valore è registrato sul file dati esterno specificato con file-designator

line-num
è il numero di linea della istruzione di programma alla quale viene trasferito il controllo della esecuzione quando il pointer del file esterno si trova alla fine del file

Azione

I valori corrispondenti alle espressioni specificate nell'istruzione dopo file-designator sono registrati nel file dati esterno identificato dal numero designatore di file nd. I valori sono registrati dalla posizione indirizzata dal pointer e nell'ordine con cui sono specificati nell'istruzione.

Se non c'è spazio sufficiente per registrare un dato sul file (sequenziale o ad accesso diretto) l'esecuzione dell'istruzione WRITE: dà una segnalazione di errore e l'esecuzione del programma è sospesa, ma i dati

specificati nell'istruzione prima del dato suddetto sono registrati nel file esterno. Se è presente l'opzione EOF il controllo della esecuzione del programma passa alla istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Note

1. Se il file è sequenziale la prima istruzione WRITE: eseguita deve essere preceduta da una istruzione SCRATCH: o APPEND:.

2. Se il file è ad accesso diretto e si vogliono registrare i dati iniziando da una parola specificata del file, l'istruzione WRITE: deve essere preceduta da una istruzione SETW:.

3. Se l'istruzione WRITE: è eseguita dopo una istruzione FILES, FILE:, SCRATCH: (solo file sequenziali) o RESTORE:, la registrazione sul file inizia dalla prima parola del file stesso.

4. nd deve essere maggiore di zero e minore od uguale al numero di file a cui il programma può accedere contemporaneamente, dichiarato con l'istruzione FILES.

5. Le costanti stringa devono essere specificate tra apici (es. "OLIVETTI").

6. Se il file a cui fa riferimento l'istruzione è ad accesso diretto, conviene non specificare come operando una espressione numerica perchè è difficile prevedere se il risultato ottenuto sarà espresso in singola o doppia precisione (quindi se occuperà 4 od 8 byte sul file esterno). Infatti, solo nel caso in cui tutti gli operandi della espressione hanno valori espressi in singola precisione, il risultato sarà espresso in singola precisione. Se invece, anche un solo operando della espressione è espresso in doppia precisione, il risultato della espressione sarà espresso in doppia precisione. Si ricorda che il risultato ritornato da una funzione numerica di sistema è sempre espresso in doppia precisione.

Esempi

1. Nel seguente esempio i file FILED1, FILED2, FILED3, FILES1, FILES2 e FILES3 hanno una lunghezza di allo-

cazione di 128 byte. Il programma registra nei file suddetti i numeri da 1 a 32, in singola precisione, che, poichè occupano 4 byte ciascuno, richiedono esattamente 128 byte. Dopo l'esecuzione del programma è stampato il contenuto dei file, meno che per i file FILES2 e FILES3 che, del resto, hanno lo stesso contenuto dei file precedenti.

```
LIST
FILE    *WRITE2

0010 JCL SA
0010 FILES FILED1;FILED2;FILED3;FILES1;FILES2;FILES3
0020 FOR I=1 TO 3 STEP 1
0030 SETW :I TO 1
0040 FOR J=1 TO 32 STEP 1
0045 LET A=J
0050 WRITE :I,A
0060 NEXT J
0070 NEXT I
0080 FOR I=4 TO 6 STEP 1
0090 SCRATCH :I
0100 FOR J=1 TO 32 STEP 1
0105 LET A=J
0110 WRITE :I,A
0120 NEXT J
0130 NEXT I
0140 END

END OF LISTING
RUN
**** FORMALLY CORRECT PROGRAM ****
            FILED1
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32

            FILED2
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32

            FILED3
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32

            FILES1
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32
```

2. Il programma sottostante registra le costanti numeriche da 1 a 16 nei file dati dell'esempio precedente. Si noti come le costanti numeriche sono registrate nei file in doppia precisione (infatti su 128 byte vengono registrati 16 dati, un dato ogni due parole, 8 byte del file). Dopo l'esecuzione del programma è stato stampato il contenuto relativo ad ogni file.

```
LIST
FILE    *WRITE1

0010 FILES FILED1;FILED2;FILED3;FILES1;FILES2;FILES3
0020 FOR I=1 TO 3 STEP 1
0030 SETW :I TO 1
0050 WRITE :I,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
0070 NEXT I
0080 FOR I=4 TO 6 STEP 1
0090 SCRATCH :I
0110 WRITE :I,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
0130 NEXT I
0140 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
            FILED1
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16

            FILED2
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16

            FILED3
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16

            FILES1
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16

            FILES2
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16

            FILES3
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16
```

3. Il programma sottostante mostra l'impiego dell'istruzione APPEND: per aggiungere al file sequenziale FILES1 la stringa "Manuale generale". Il file viene poi letto dallo stesso programma.

```
LIST
FILE    *WRITE4

0010 FILES FILES1
0020 APPEND :1
0030 WRITE :1,"Manuale generale"
0040 RESTORE :1
0050 FOR I=1 TO 10 STEP 1
0060 READ :1,A$ EOF 90
0070 PRINT A$
0080 NEXT I
0090 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
OLIVETTI
OLIVETTI
Manuale generale
```

4. La routine sottostante mostra come per leggere un dato in un file ad accesso diretto, dopo averlo registrato, si deve spostare con l'istruzione SETW: il relativo pointer all'inizio della parola da cui è stato registrato.

```
LIST
FILE    *WRITE5

0005 DCL 20A$
0010 FILES FILED1
0020 WRITE :1,"Primo dato del file"
0030 SETW :1 TO 1
0040 READ :1,A$
0050 PRINT A$
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Primo dato del file
```

5. In questo programma si può vedere come agisce l'opzione EOF in una istruzione riferita ad un file ad accesso diretto (FILED1) ed in un file sequenziale (FILES1).

```
LIST
FILE    *WRITE6

0010 FILES FILED1;FILES1
0020 SCRATCH :2
0030 LET I=0
0040 LET I=I+1
0050 IF I>2 THEN 90
0060 PRINT
0070 WRITE :I,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19 EOF 270
0090 SETW :1 TO 1
0100 RESTORE :2
0120 LET I=0
0130 LET I=I+1
0140 IF I>2 THEN 290
0150 PRINT
0160 PRINT
0170 PRINT "Ecco il contenuto del file numero designatore";I;":"
0180 FOR J=1 TO 17 STEP 1
0190 READ :I,A EOF 250
0200 PRINT A,
0210 NEXT J
0240 GOTO 290
0250 PRINT "Nel file con numero designatore";I;"non vi sono piu` dati."
0260 GOTO 130
0270 PRINT "Nel file con numero designatore";I;"non ci stanno piu` dati!"
0280 GOTO 40
0290 END

END OF LISTING


RUN

Nel file con numero designatore 1 non ci stanno piu` dati!

Nel file con numero designatore 2 non ci stanno piu` dati!


Ecco il contenuto del file numero designatore 1 :
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16            Nel file con numero designatore 1 non vi sono piu` dati.


Ecco il contenuto del file numero designatore 2 :
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16            Nel file con numero designatore 2 non vi sono piu` dati
```

6. Il programma sottostante mostra come si può aggiornare un dato in un file ad accesso diretto (FILED1) prelevando i dati dallo stesso file e da un altro file dati. Per vedere da quali parole dei file sono prelevati i dati e come il file FILED1 si modifica è stato stampato il contenuto relativo ai suddetti file.

```
EXEC FLPRINT,FILED1
 1                2                3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16
END OF PRINT
EXEC FLPRINT,FILED2
 1                2                3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16               17               18               19               20
 21               22               23               24               25
 26               27               28               29               30
 31               32
END OF PRINT
EXEC FLPRINT,FILED3
 1                2                3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16               17               18               19               20
 21               22               23               24               25
 26               27               28               29               30
 31               32
END OF PRINT


OLD *WRITE7
LIST
FILE    *WRITE7


0010 FILES FILED1,FILED2,FILED3
0020 SETW :2 TO 10
0030 SETW :3 TO 5
0040 SETW :1 TO 3
0050 READ :1,A
0060 READ :2,B
0070 READ :3,C
0080 LET A=A*B*C
0090 SETW :1 TO 3
0100 WRITE :1,A
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****


EXEC FLPRINT,FILED1
 1                100              3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16
END OF PRINT
```

Il linguaggio BASIC P6066 offre la possibilità di elaborare le matrici numeriche con le stesse regole del calcolo matriciale. Nel seguente paragrafo sono descritte le istruzioni del linguaggio BASIC che permettono di eseguire le elaborazioni suddette. Per poter eseguire queste istruzioni si deve aver prima eseguito un comando OPTIONS specificando l'operando MAT. Le istruzioni di elaborazione delle matrici sono denominate istruzioni di assegnazione perchè il risultato del calcolo è assegnato ad una matrice specificata; esse sono:

```
MAT matrix = matrix
MAT matrix = matrix + matrix
MAT matrix = matrix - matrix
MAT matrix = (num-exp) * matrix
MAT matrix = matrix * matrix
MAT matrix = CON  (num-exp, num-exp)
MAT matrix = IDN  (num-exp, num-exp)
MAT matrix = INV  (matrix)
MAT matrix = TNR  (matrix)
MAT matrix = ZER  (num-exp, num-exp)
```

Nelle suddette istruzioni la matrice è considerata un elemento sintattico; infatti il campo specificato con "matrix" deve essere sostituito con il nome di una matrice. Vedremo, caso per caso, se lo stesso nome di matrice può essere espresso a destra ed a sinistra del segno uguale e se la matrice deve avere le due dimensioni uguali (matrice quadrata) o può averle diverse (matrice rettangolare). Per quanto riguarda lo spazio di memoria principale richiesto da una matrice si veda l'appendice F.

Come già visto, per una matrice si distinguono le dimensioni di allocazione da quelle attuali. Le dimensioni di allocazione sono dichiarate esplicitamente con una istruzione DIM e stabiliscono rispettivamente il numero di righe e di colonne riservate per la matrice (in ultima analisi il numero di elementi che la

possono costituire). Se nell'ambito del programma non vi sono istruzioni DIM riferite ad una matrice, s'intende che per essa sono riservate 10 righe e 10 colonne. Le dimensioni attuali sono quelle effettivamente utilizzate dalla matrice, ossia le righe e colonne che realmente utilizza.

Alcune istruzioni non si limitano ad assegnare agli elementi di una matrice dei nuovi valori come risultato della elaborazione definita dal rispettivo algoritmo (ad es. moltiplicazione, righe per colonne, di due matrici etc.), ma definiscono delle nuove dimensioni attuali per la matrice suddetta, il cui prodotto sarà sempre minore o uguale al prodotto delle dimensioni di allocazione. Per quando riguarda le assegnazioni di valori agli elementi di una matrice, si possono verificare le seguenti situazioni a seconda del tipo di precisione dichiarata per la matrice:

1. Nel caso che la matrice sia dichiarata in singola precisione:

   - se ad un elemento della matrice viene assegnato un valore in virgola mobile la cui mantissa è al di fuori del campo di rappresentazione, ma il cui esponente rientra nel campo suddetto, il valore numerico viene troncato alle prime 6 cifre significative

   - se ad un elemento della matrice è assegnato un valore il cui esponente è al di fuori del campo di rappresentazione, il valore è eguagliato a + 9.99999E63 (se il valore era positivo) od a -9.999999E63 (se il valore era negativo)

   - se ad un elemento di una matrice è assegnato un valore compreso nella zona di UNDERFLOW per la singola precisione, il valore è uguagliato a <u>zero</u>

2. Nel caso che la matrice sia dichiarata in doppia precisione (dichiarazione implicita):

   - se ad un elemento della matrice è assegnato un valore compreso nella zona di OVERFLOW, il valore è uguagliato a 9.999999999999E99 (se il valore era positivo) oppure a - 9.999999999999E99 (se il valore era negativo)

- se ad un elemento di una matrice è assegnato un valore compreso nella zona di UNDERFLOW, il valore è uguagliato a zero

Quando si verifica ognuno dei casi suddetti (meno il primo caso del punto 1.) il sistema commuta nello stato di debugging dopo aver effettuato l'assegnazione specificata caso per caso. L'utente può far continuare l'esecuzione del programma premendo CONTINUE oppure far terminare l'esecuzione stessa premendo il tasto BREAK. (Il sistema commuta nello stato comandi). Si noti inoltre che tutte le istruzioni suddette possono far riferimento, come casi particolari, a vettori. Infatti, se una delle due dimensioni di una matrice è dichiarata uguale ad 1, la matrice è in sostanza un vettore . Nell'ultima parte del paragrafo sono descritte le istruzioni che permettono di assegnare agli elementi delle matrici dei valori da tastiera (MAT INPUT), da file dati interno (MAT READ), da file dati esterno (MAT READ:), le istruzioni che permettono di stampare i valori degli elementi delle matrici (MAT PRINT e MAT PRINT USING) e di registrare detti valori su file dati esterno (MAT WRITE:).

Le istruzioni suddette possono avere come operandi non solo delle matrici numeriche, ma anche delle variabili multiple di tipo stringa. Con eccezione dell'istruzione INPUT, le altre istruzioni possono avere più di un operando ed in questo caso si possono avere variabili multiple di diverso tipo, numeriche o stringa, contemporaneamente. Anche le suddette istruzioni permettono (meno l'istruzione MAT WRITE:) di modificare le dimensioni attuali delle variabili multiple a cui si riferiscono. Negli esempi riportati nella descrizione delle istruzioni che segue sono utilizzate le istruzioni MAT INPUT, MAT PRINT e MAT READ, per cui ad una prima lettura del manuale si consiglia di leggere prima le descrizioni relative a tali istruzioni, questo faciliterà la comprensione degli esempi suddetti.

Istruzione MAT ... =

Funzione

Assegna i valori degli elementi di una matrice agli elementi di un'altra matrice.

Formato

MAT matrix = matrix

dove:
matrix
indica il nome di una matrice numerica quadrata o rettangolare.

Azione

Ogni valore di ogni elemento della matrice specificata sulla destra del segno uguale è assegnato al corrispondente elemento della matrice alla sinistra del segno uguale.

La matrice alla sinistra del segno uguale assume le dimensioni attuali della matrice a destra del segno uguale.

Nota

Il prodotto delle dimensioni di allocazione della matrice a sinistra del segno uguale deve essere maggiore od uguale al prodotto delle dimensioni attuali della matrice a destra del segno uguale.

Esempi

1. La routine sottostante richiede (istruzione 20) nove dati numerici da tastiera da assegnare agli elementi della matrice A (avente dimensioni di allocazione di 10 x 10 elementi) che viene ad assumere dimensioni attuali pari a 3 x 3 elementi. L'istruzione 30 assegna quindi i valori degli elementi della matrice A agli elementi della matrice B che assume dimensioni attuali di 3 x 3 elementi. Dalla stampa prodotta in seguito all'esecuzione della routine si

può vedere il risultato ottenuto.

```
LIST
FILE


0010 DISP "Introduci i valori
0020 MAT INPUT A(3,3)
0030 MAT B=A
0040 MAT PRINT A;
0050 PRINT
0060 MAT PRINT B;
0070 END

END OF LISTING


RUN
Introduci i valori                    ?
1,2,3,4,5,6,7,8,9
 1  2  3
 4  5  6
 7  8  9

 1  2  3
 4  5  6
 7  8  9
```

2. La routine sottostante dichiara, istruzione 10, per la matrice A e per la matrice B dimensioni di allocazione rispettivamente di 3 x 3 e 2 x 3 elementi. L'istruzione 30 richiede 9 dati numerici da tastiera da assegnare alla matrice A, mentre l'istruzione 70 assegna agli elementi della matrice B i valori numerici specificati nella istruzione 140. Infine l'istruzione 80 assegna agli elementi della matrice A i valori degli elementi della matrice B e la matrice A assume le dimensioni attuali di 2 x 3 elementi. Dalla stampa prodotta con l'esecuzione del programma si può vedere il risultato ottenuto.

```
LIST
FILE    *MAT2


0010 DIM A(3,3),B(2,3)
0020 DISP "INTRODUCI I VALORI PER A()                ";
0030 MAT INPUT A
0040 PRINT "LA MATRICE A HA I VALORI:"
0050 MAT PRINT A;
0060 PRINT
0070 MAT READ B
0080 MAT A=B
0090 PRINT "LA MATRICE B HA I VALORI:"
0100 MAT PRINT B;
0110 PRINT
```

```
0120 PRINT "ORA LA MATRICE A HA I VALORI:"
0130 MAT PRINT A;
0140 DATA 1,2,3,4,5,6
0150 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
INTRODUCI I VALORI PER A()                    ?
9,9,9,9,9,9,9,9,9
LA MATRICE A HA I VALORI:
 9  9  9
 9  9  9
 9  9  9

LA MATRICE B HA I VALORI:
 1  2  3
 4  5  6

ORA LA MATRICE A HA I VALORI:
 1  2  3
 4  5  6
```

Istruzione MAT ... +

Funzione

Esegue l'operazione di addizione tra due matrici e ne assegna il risultato alla matrice specificata prima del segno uguale.

Formato

MAT matrix = matrix + matrix

dove:
matrix
indica il nome di una matrice numerica quadrata rettangolare.

Azione

I valori degli elementi corrispondenti delle due matrici numeriche a destra del segno uguale sono addizionati ed i risultati della operazione sono assegnati ai corrispondenti elementi della matrice specificata a sinistra del segno uguale.

La matrice a sinistra del segno uguale assume le dimensioni attuali delle matrici a destra del segno uguale.

Note

1. La somma tra i valori degli elementi delle matrici a destra del segno uguale è una somma algebrica.

2. Le due matrici a destra del segno uguale devono avere le stesse dimensioni attuali.

3. Il prodotto delle dimensioni di allocazione della matrice a sinistra del segno uguale deve essere maggiore od uguale al prodotto delle dimensioni attuali delle due matrici a destra del segno uguale.

4. La matrice che compare a sinistra del segno uguale può comparire anche a destra del segno uguale una

o due volte.

Esempi

1. La seguente routine calcola la somma tra due matrici B e C ed assegna il risultato ad una terza matrice A.

```
LIST
FILE    *MAT4

0010 DIM A(3,2),B(3,2),C(3,2)
0020 DISP "Introduci i valori per B          ";
0030 MAT INPUT B
0040 DISP "Introduci i valori per C          ";
0050 MAT INPUT C
0060 MAT A=B+C
0070 PRINT "I valori della matrice B sono:"
0080 MAT PRINT B;
0090 PRINT
0100 PRINT "I valori della matrice C sono:"
0110 MAT PRINT C;
0120 PRINT "I valori della matrice A sono:"
0130 MAT PRINT A;
0140 END

END OF LISTING


RUN
Introduci i valori per B          ?
1,2,3,4,5,6
Introduci i valori per C          ?
1,2,3,4,5,6
I valori della matrice B sono:
 1  2
 3  4
 5  6

I valori della matrice C sono:
 1  2
 3  4
 5  6
I valori della matrice A sono:
 2  4
 6  8
 10  12
RUN
Introduci i valori per B          ?
-25.55,15,-12,+7,-10,150
Introduci i valori per C          ?
5.12,-7,-12,+11.55,+15.8,-150
I valori della matrice B sono:
 5.12 -7
-12  11.55
 15.8 -150
I valori della matrice C sono:
-25.55  15
-12  7
-10  150
I valori della matrice A sono:
-20.43  8
-24  18.55
 5.8  0
```

2. Nell'esempio seguente si può vedere come non sia possibile eseguire l'addizione di due matrici che non abbiano lo stesso numero di righe e lo stesso numero di colonne, come dimensioni attuali. L'esecuzione della istruzione 60, infatti, produce una segnalazione di errore, come si può vedere. Premendo BREAK il sistema commuta nello stato di comandi. Con il comando FETCH (FET 10) si richiama nel buffer di tastiera l'istruzione 10 e la si modifica ponendo B (3,2) invece di B (2,3), per cui una successiva esecuzione (comando RUN) produce la stampa dei valori di B (istruzione 80) di C (istruzione 110) e di A (istruzione 130).

```
LIST
FILE    *MAT5

0010 DIM A(4,4),B(3,2),C(2,3)
0020 DISP "Introduci i valori per B      ";
0030 MAT INPUT B
0040 DISP "Introduci i valori per C      ";
0050 MAT INPUT C
0060 MAT A=B+C
0070 PRINT "I valori della matrice B sono:"
0080 MAT PRINT B;
0090 PRINT
0100 PRINT "I valori della matrice C sono:"
0110 MAT PRINT C;
0120 PRINT "I valori della matrice A sono:"
0130 MAT PRINT A;
0140 END

END OF LISTING


RUN
Introduci i valori per B      ?
1,2,3,4,5,6
Introduci i valori per C      ?
1,2,3,4,5,6
ERROR 73  IN LINE 60
FET 10
0010 DIM A(4,4),B(3,2),C(2,3)
0010 DIM A(4,4),B(3,2),C(3,2)
RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per B      ?
1,2,3,4,5,6
Introduci i valori per C      ?
4,4,4,4,4,4
I valori della matrice B sono:
 1  2
 3  4
 5  6

I valori della matrice C sono:
 4  4
 4  4
 4  4
I valori della matrice A sono:
 5  6
 7  8
 9  10
```

3. In questo esempio si dichiara la matrice A (istruzione 5) in singola precisione. Al primo elemento della matrice B si assegna da tastiera un valore con esponente nella zona di OVERFLOW per la rappresentazione in singola precisione (10E78), istruzione 30, mentre all'ultimo elemento della matrice C si assegna un valore nella zona di UNDERFLOW per la rappresentazione in singola precisione (1E-85). Quando viene eseguita l'istruzione 60 il sistema assegna al primo elemento della matrice A il valore 9.99999E63 ed all'ultimo elemento della stessa matrice il valore 6 (a 6 viene sommato lo zero) come si può vedere dalla stampa prodotta con l'istruzione 130. Viene visualizzato l'errore sottoriportato ed il sistema è nello stato di debugging. Premendo il tasto CONTINUE l'esecuzione prosegue e vengono prodotte le stampe sottoriportate.

```
LIST
FILE    *MAT6

0005 DCL S(A())
0010 DIM A(4,4),B(3,2),C(3,2)
0020 DISP "Introduci i valori per B          ";
0030 MAT INPUT B
0040 DISP "Introduci i valori per C          ";
0050 MAT INPUT C
0060 MAT A=B+C
0070 PRINT "I valori della matrice B sono:"
0080 MAT PRINT B;
0090 PRINT
0100 PRINT "I valori della matrice C sono:"
0110 MAT PRINT C;
0120 PRINT "I valori della matrice A sono:"
0130 MAT PRINT A;
0140 END

END OF LISTING

RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per B          ?
10E78,2,3,4,5,6
Introduci i valori per C          ?
1,2,3,4,5,1E-85
ERROR 3   IN LINE 60
I valori della matrice B sono:
 1.0000000E+79  2
 3  4
 5  6

I valori della matrice C sono:
 1  2
 3  4
 5  1.0000000E-85
I valori della matrice A sono:
 9.9999900E+63  4
 6  8
 10  6
```

Istruzione MAT ... –

Funzione

Esegue l'operazione di sottrazione tra due matrici e ne assegna il risultato ad una matrice specificata.

Formato

MAT matrix = matrix – matrix

dove:
matrix
indica il nome di una matrice numerica quadrata o rettangolare.

Azione

Ai valori degli elementi della prima matrice a destra del segno uguale sono sottratti i valori degli elementi corrispondenti della seconda matrice a destra del segno uguale; i risultati sono assegnati ai corrispondenti elementi della matrice specificata a sinistra del segno uguale.

La matrice a sinistra del segno uguale assume le dimensioni attuali delle matrici a destra del segno uguale.

Note

1. La differenza tra i valori degli elementi delle matrici a destra del segno uguale è una differenza algebrica.

2. Le due matrici a destra del segno uguale devono avere le stesse dimensioni attuali.

3. Il prodotto delle dimensioni di allocazione della matrice a sinistra del segno uguale deve essere maggiore od uguale al prodotto delle dimensioni attuali delle due matrici a destra del segno uguale.

4. La matrice che compare a sinistra del segno uguale

può comparire anche a destra del segno uguale una o due volte.

Esempio

La routine sottostante calcola la differenza algebrica tra due matrici B e C.

```
LIST
FILE    *MAT7

0010 DIM A(25,35),B(5,5),C(5,5)
0020 DISP "Introduci i valori per B        ";
0030 MAT INPUT B
0040 DISP "Introduci i valori per C         ";
0050 MAT INPUT C
0060 MAT A=B-C
0070 PRINT
0080 PRINT "I valori di B sono:"
0090 MAT PRINT B;
0100 PRINT
0110 PRINT "I valori di C sono:"
0120 MAT PRINT C;
0130 PRINT
0140 PRINT "I valori di A sono:"
0150 MAT PRINT A;
0160 END

END OF LISTING


RUN
Introduci i valori per B         ?
1,2,3,6,5,4,7,9,8,1,2,6,3,5,8,9,7,4,1,2,-6,-9,-8,-9,-5
Introduci i valori per C          ?
-9,-5,-8,-9,5,-2,-6,5,8,9,5,5,2,6,6,9,3,5,8,-9,-5,-6,-9,-3,-5

I valori di B sono:
 1  2  3  6  5
 4  7  9  8  1
 2  6  3  5  8
 9  7  4  1  2
-6 -9 -8 -9 -5

I valori di C sono:
-9 -5 -8 -9  5
-2 -6  5  8  9
 5  5  2  6  6
 9  3  5  8 -9
-5 -6 -9 -3 -5

I valori di A sono
 10  7  11  15  0
 6  13  4  0 -8
-3  1  1 -1  2
 0  4 -1 -7  11
-1 -3  1 -6. 0
```

MAT...* (moltip scalare)

Istruzione
MAT...* Scalare

Funzione

Moltiplica ogni elemento di una matrice per il valore di una espressione numerica e ne assegna il risultato ad un'altra matrice specificata.

Formato

MAT matrix = (num-exp) * matrix

dove:

num-exp
indica una espressione numerica che viene moltiplicata per ogni valore degli elementi della matrice specificata a destra del segno uguale

matrix
indica il nome di una matrice numerica quadrata o rettangolare.

Azione

L'espressione numerica è eseguita ed il valore ottenuto è moltiplicato per il valore di ogni elemento della matrice numerica a destra del segno uguale. Il risultato ottenuto è assegnato al corrispondente elemento della matrice a sinistra del segno uguale.

La matrice a sinistra del segno uguale assume le dimensioni attuali della matrice a destra del segno uguale.

Note

1. La matrice a sinistra del segno uguale non deve avere il prodotto delle dimensioni di allocazione minore del prodotto delle dimensioni attuali della matrice a destra del segno uguale.

2. La matrice che compare a sinistra del segno uguale può comparire anche a destra del segno uguale.

Esempio

La routine seguente esegue il prodotto scalare di una variabile numerica con una matrice, istruzione 40 e di una costante numerica con una matrice, istruzione 130.

```
LIST
FILE    *MAT8

0010 MAT READ A(3,4)
0020 DISP "Introduci moltiplicatore di A      ";
0030 INPUT B
0040 MAT C=(B)*A
0050 PRINT
0060 PRINT "Il valore di  B e`:";B
0070 PRINT
0080 PRINT "I valori di A sono:"
0090 MAT PRINT A;
0100 PRINT
0110 PRINT "I valori di C sono:"
0120 MAT PRINT C;
0130 MAT C=(5)*C
0140 PRINT
0150 PRINT "Ora i valori di C sono:"
0160 MAT PRINT C;
0170 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,1,5,16
0180 END

END OF LISTING


RUN
Introduci moltiplicatore di A      ?
-10

Il valore di  B e`:-10

I valori di A sono:
 1  2  3  4
 5  6  7  8
 9  10  11  12

I valori di C sono:
-10 -20 -30 -40
-50 -60 -70 -80
-90 -100 -110 -120

Ora i valori di C sono:
-50 -100 -150 -200
-250 -300 -350 -400
-450 -500 -550 -600
```

Istruzione MAT ... *

Funzione

Esegue il prodotto, righe per colonne, tra due matrici e ne assegna il risultato ad una matrice specificata.

Formato

MAT matrix = matrix * matrix

dove:
matrix
indica il nome di una matrice numerica quadrata o rettangolare.

Azione

Ogni elemento di una riga (i) della prima matrice numerica a destra del segno uguale è moltiplicato per l'elemento che compare nello stesso ordine in una colonna (j) della seconda matrice numerica a destra del segno uguale.

I prodotti così ottenuti sono sommati; il risultato ottenuto è assegnato all'elemento della matrice a sinistra del segno uguale che si trova nella riga i e nella colonna j.

Se si moltiplica una matrice A di dimensioni attuali (p,m) con una matrice B di dimensioni attuali (m,n) si ottiene una matrice C di dimensioni attuali (p,n) tale che per i = 1, 2, ..., p e per j = 1, 2, ..., n:

$$c_{i,j} = \sum_{k=1}^{m} a_{i,k} * b_{k,j}$$

dove $c_{i,j}$ è un elemento generico della matrice C e $a_{i,k}$ e $b_{k,j}$ sono elementi rispettivamente della matrice A e B.

Note

1. La matrice numerica che compare a sinistra del segno uguale non deve comparire anche a destra del segno uguale.

2. Il prodotto delle dimensioni di allocazione della matrice a sinistra del segno uguale deve essere maggiore od uguale al prodotto del numero attuale di righe della prima matrice a destra del segno uguale per il numero attuale di colonne della seconda matrice a destra del segno uguale.

3. Il numero di colonne della prima matrice a destra del segno uguale deve essere uguale al numero di righe della seconda matrice a destra del segno uguale.

Esempio

La routine sottostante calcola il prodotto, righe per colonne, della matrice A con la matrice B. Con le stampe prodotte durante l'esecuzione della routine si possono vedere i valori assegnati alla matrice A, alla matrice B, ed alla matrice C.

```
LIST
FILE    *MAT9

0010 REM Ecco il prodotto righe per colonne tra la matrice A e la matrice B
0020 DIM A(2,3),B(3,2)
0030 DISP "Introduci i valori per A          ";
0040 MAT INPUT A
0050 DISP "Introduci i valori per B          ";
0060 MAT INPUT B
0070 MAT C=A*B
0080 PRINT
0090 PRINT "I valori di A sono:"
0100 MAT PRINT A;
0110 PRINT
0120 PRINT "I valori di B sono:"
0130 MAT PRINT B;
0140 PRINT
0160 PRINT "I valori di C sono:"
0170 MAT PRINT C;
0180 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per A          ?
1,-5,+8,-3,4,-8
Introduci i valori per B          ?
0,-5,-4,4,8,+6

I valori di A sono:
 1 -5  8
-3  4 -8

I valori di B sono:
 0 -5
-4  4
 8  6

I valori di C sono:
 84  23
-80 -17
```

Istruzione MAT ...CON

Funzione

Assegna il valore uno ad ogni elemento di una matrice.

Formato

MAT matrix = CON [**(num-exp, num-exp)**]

dove:

matrix
indica il nome di una matrice numerica quadrata o rettangolare

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, rappresenta la nuova dimensione attuale della matrice specificata a sinistra del segno uguale.

Azione

A tutti gli elementi della matrice numerica a sinistra del segno uguale è assegnato il valore 1.

Se è presente la parte opzionale, la matrice numerica a sinistra del segno uguale assume le dimensioni attuali pari ai valori ottenuti eseguendo le due espressioni numeriche indicate tra parentesi (che sono arrotondate all'intero più prossimo). Se la parte opzionale non è specificata le dimensioni attuali della matrice non sono modificate.

Nota

Il prodotto delle dimensioni di allocazione della matrice a sinistra del segno uguale deve essere maggiore od uguale al prodotto dei valori, arrotondati all'intero più prossimo, ottenuti eseguendo le due espressioni numeriche della parte opzionale.

Esempio

La routine sottostante pone in evidenza come l'istruzione che assegna la costante uno agli elementi di una matrice ne permette anche il ridimensionamento. L'istruzione 70 assegna la costante uno a tutti gle elementi allocati in memoria principale per la matrice Z, come si può vedere con la stampa prodotta dall'istruzione 100. L'istruzione 110 assegna a sei elementi delle prime cinque righe della matrice B la costante uno, come si può vedere dalla stampa prodotta con l'istruzione 140.

```
LIST
FILE    *MAT10

0010 DIM Z(5,5)
0020 DISP "Introduci i valori per B          ";
0030 MAT INPUT B(1,5)
0040 PRINT
0050 PRINT "I valori della matrice B sono:"
0060 MAT PRINT B;
0070 MAT Z=CON
0080 PRINT
0090 PRINT "I valori della matrice Z sono:"
0100 MAT PRINT Z;
0110 MAT B=CON(5,6)
0120 PRINT
0130 PRINT "Ora i valori della matrice B sono:"
0140 MAT PRINT B;
0150 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per B          ?
1,2,3,4,5

I valori della matrice B sono:
 1  2  3  4  5

I valori della matrice Z sono:
 1  1  1  1  1
 1  1  1  1  1
 1  1  1  1  1
 1  1  1  1  1
 1  1  1  1  1

Ora i valori della matrice B sono:
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
```

Istruzione MAT ... IDN

Funzione

Assegna il valore uno a tutti gli elementi della diagonale principale di una matrice quadrata ed il valore zero a tutti gli altri elementi della matrice.

Formato

MAT matrix = IDN |(num-exp, num-exp)|

dove:

matrix
indica il nome di una matrice numerica quadrata

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, rappresenta la nuova dimensione attuale della matrice specificata a sinistra del segno uguale.

Azione

Agli elementi con indici uguali, della matrice numerica quadrata indicata a sinistra del segno uguale, è assegnato il valore 1.

A tutti gli altri elementi della matrice suddetta è assegnato il valore Ø.

Se è presente la parte opzionale le espressioni numeriche sono eseguite ed i valori ottenuti sono arrotondati all'intero più prossimo. La matrice numerica quadrata indicata nella istruzione assume le dimensioni attuali specificate dai valori delle espressioni numeriche racchiuse tra parentesi.

Se la parte opzionale non è specificata le dimensioni attuali della matrice specificata non sono modificate.

Note

1. I valori ottenuti eseguendo le due espressioni numeriche indicate nella parte opzionale, arrotondati

all'intero più prossimo, devono essere uguali tra loro e maggiori di zero.

2. Il prodotto delle dimensioni di allocazione della matrice numerica quadrata deve essere maggiore od uguale al prodotto dei valori ottenuti dalle espressioni numeriche specificate tra parentesi.

Esempio

La routine sottostante mostra come anche l'istruzione che assegna ad una matrice i valori della matrice identità permette di ridimensionare le dimensioni attuali. L'istruzione 10 assegna alla matrice A la matrice identità secondo le dimensioni di allocazione (10 x 10). L'istruzione 50 assegna alla stessa matrice A la matrice identità, ma secondo le nuove dimensioni attuali (4,4).

```
LIST
FILE    +MAT11

0010 MAT A=IDN
0020 PRINT
0030 PRINT "I valori di A sono:"
0040 MAT PRINT A;
0050 MAT A=IDN(4,4)
0060 PRINT
0070 PRINT "Ora i valori di A sono:"
0080 MAT PRINT A;
0090 END

END OF LISTING.


RUN
**** FORMALLY CORRECT PROGRAM ****

I valori di A sono:
 1  0  0  0  0  0  0  0  0  0
 0  1  0  0  0  0  0  0  0  0
 0  0  1  0  0  0  0  0  0  0
 0  0  0  1  0  0  0  0  0  0
 0  0  0  0  1  0  0  0  0  0
 0  0  0  0  0  1  0  0  0  0
 0  0  0  0  0  0  1  0  0  0
 0  0  0  0  0  0  0  1  0  0
 0  0  0  0  0  0  0  0  1  0
 0  0  0  0  0  0  0  0  0  1

Ora i valori di A sono:
 1  0  0  0
 0  1  0  0
 0  0  1  0
 0  0  0  1
```

Istruzione MAT ... INV

Funzione

Calcola la matrice inversa di una matrice quadrata e la assegna ad una matrice specificata.

Formato

MAT matrix = INV (matrix)

dove:
matrix
indica il nome di una matrice numerica.

Azione

La matrice inversa della matrice numerica quadrata indicata alla destra del segno uguale viene assegnata, elemento per elemento, alla matrice numerica indicata alla sinistra del segno uguale.

Data una matrice numerica quadrata M di dimensioni (m,m) la matrice inversa N, se esiste, è la matrice di eguali dimensioni tali che:

$$M * N = N * M = I$$

dove I è una matrice identità.

Non tutte le matrici hanno una matrice inversa, infatti le matrici con determinante uguale a zero non hanno una matrice inversa.

La matrice a sinistra del segno uguale assume le stesse dimensioni attuali della matrice a destra del segno uguale.

Note

1. Il prodotto delle dimensioni di allocazione della matrice a sinistra del segno uguale deve essere maggiore od uguale al prodotto delle dimensioni attuali della matrice a destra del segno uguale.

2. Il calcolo della inversione di una matrice fornisce anche il valore del suo determinante, che è fornito al programma utilizzando la funzione di sistema DET. Quindi, se in un programma compare l'istruzione 70 MAT B = INV (A) ed in una istruzione successiva la funzione DET, quest'ultima ritorna il valore del determinante della matrice A.

3. La matrice a destra del segno uguale deve essere una matrice quadrata.

4. La matrice a sinistra del segno uguale può essere specificata anche a destra del segno uguale.

5. Se il determinante della matrice a destra del segno uguale è zero, è visualizzato un errore di tipo recuperabile ed il sistema è nello stato di debugging; premendo CONTINUE si può continuare l'esecuzione del programma.

Esempio

Il seguente programma calcola l'inversa della matrice A e della matrice B, stampa quindi i valori ottenuti per le due matrici ed il valore dei rispettivi determinanti. Sono riportate due esecuzioni del programma. Durante la prima esecuzione i valori forniti da tastiera sono tali da rendere uguale a zero il determinante della matrice A ed il determinante della matrice B, per cui l'esecuzione del programma è interrotta due volte ed è emesso un messaggio di errore recuperabile. Premendo il tasto CONTINUE l'esecuzione riprende ogni volta e la seconda volta continua fino al termine.

```
LIST
FILE    +MAT12

0010 DISP "Introduci i valori per A      ";
0020 MAT INPUT A(5,5)
0030 PRINT
0040 PRINT "I valori di A sono:"
0050 MAT PRINT A;
0060 PRINT
0070 MAT B=INV(A)
0080 PRINT "I valori di B sono"
0090 MAT PRINT B;
0100 PRINT
0110 PRINT "Il DETERMINANTE di A e':";DET
0120 MAT B=INV(B)
0130 PRINT
0140 PRINT "Il DETERMINANTE di B e':";DET
0150 PRINT
0160 PRINT "Ora il valore di B e':"
```

```
0170 MAT PRINT B;
0180 END

END OF LISTING
RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per A      ?
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5

I valori di A sono:
 5  5  5  5  5
 5  5  5  5  5
 5  5  5  5  5
 5  5  5  5  5
 5  5  5  5  5

ERROR 13  IN LINE 70
I valori di B sono
 9.9999999E+99  0  0  0 -9.9999999E+99
 0  9.9999999E+99  0  0 -9.9999999E+99
 0  0  9.9999999E+99  0 -9.9999999E+99
 0  0  0  9.9999999E+99 -9.9999999E+99
-9.9999999E+99 -9.9999999E+99 -9.9999999E+99 -9.9999999E+99  9.9999999E+99

Il DETERMINANTE di A e': 0
ERROR 4   IN LINE 120

Il DETERMINANTE di B e': 9.9999999E+99

Ora il valore di B e':
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
RUN
Introduci i valori per A      ?
2,3,6,5,4,6,5,2,1,4,8,7,9,6,3,2,5,8,4,6,5,9,7,3,2

I valori di A sono:
 2  3  6  5  4
 6  5  2  1  4
 8  7  9  6  3
 2  5  8  4  6
 5  9  7  3  2

I valori di B sono
-.19293478  7.3369565E-02  .20652174  2.7173913E-02 -.15217391
 .24565217  5.8695652E-02 -.23478261 -.17826087  .27826087
-.47445652 -.18858696  .25434783  .35978261 -.13478261
 .65434783  4.1304348E-02 -.16521739 -.42173913  .12173913
 5.5978261E-02  .15054348 -.10217391  .10760870 -8.2608696E-02

Il DETERMINANTE di A e':-1840.0000

Il DETERMINANTE di B e':-5.4347826E-04

Ora il valore di B e':
 2.0000000  3.0000000  6.0000000  5.0000000  4.0000000
 6.0000000  5.0000000  2.0000000  1.0000000  4.0000000
 8.0000000  7.0000000  9.0000000  6.0000000  3.0000000
 2.0000000  5.0000000  8.0000000  4.0000000  6.0000000
 5.0000000  9.0000000  7.0000000  3.0000000  2.0000000
```

Istruzione MAT ... TRN

Funzione

Assegna ad una matrice specificata gli elementi di un'altra matrice scambiando tra loro le righe con le colonne.

Formato

MAT matrix = TRN (matrix)

dove:
matrix
indica il nome di una matrice numerica quadrata e rettangolare.

Azione

Le righe e le colonne della matrice a destra del segno uguale sono scambiate tra loro e la nuova matrice così costruita è assegnata alla matrice indicata a sinistra del segno uguale.

I valori della colonna x della matrice a destra del segno uguale coincidono con i valori della riga x della matrice a sinistra del segno uguale. I valori della riga y della matrice a destra del segno uguale coincidono con i valori della colonna y della matrice a sinistra del segno uguale.

Se la matrice a destra del segno uguale ha dimensioni attuali (m,n), la matrice a sinistra del segno uguale assume dimensioni attuali (n,m).

Note

1. Il prodotto delle dimensioni di allocazione della matrice a sinistra del segno uguale deve essere maggiore od uguale al prodotto delle dimensioni attuali della matrice a destra del segno uguale.

2. Non si deve avere la stessa matrice da entrambi i lati del segno uguale.

Esempio

La routine sottostante assegna alla matrice A i valori della matrice B scambiando tra loro le righe con le colonne.

```
LIST
FILE    +MAT13

0010 DISP "Introduci i valori per B            ";
0020 MAT INPUT B(4,3)
0030 MAT A=TRN(B)
0040 PRINT
0050 PRINT "I valori di B sono:"
0060 MAT PRINT B;
0070 PRINT
0080 PRINT "I valori di A sono:"
0090 MAT PRINT A;
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per B            ?
10,11,12,21,22,23,31,32,33,41,42,43

I valori di B sono:
 10  11  12
 21  22  23
 31  32  33
 41  42  43

I valori di A sono:
 10  21  31  41
 11  22  32  42
 12  23  33  43
```

Istruzione MAT ... ZER

Funzione

Assegna il valore zero a tutti gli elementi di una matrice.

Formato

MAT matrix = ZER [(num-exp, num-exp)]

dove:

matrix
indica il nome di una matrice numerica quadrata o rettangolare

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, rappresenta la nuova dimensione attuale della matrice specificata a sinistra del segno uguale.

Azione

A tutti gli elementi della matrice numerica indicata nella istruzione è assegnato il valore zero.

Se è indicata la parte opzionale, le espressioni numeriche sono eseguite ed i valori ottenuti m ed n arrotondati all'intero più prossimo costituiscono le nuove dimensioni attuali della matrice specificata.

Se la parte opzionale non è specificata le dimensioni attuali della matrice specificata non sono modificate.

Nota

Il prodotto delle dimensioni di allocazione della matrice specificata nell'istruzione deve essere maggiore od uguale al prodotto m*n.

Esempio

Il seguente programma assegna il valore zero agli elementi delle prime tre righe e quattro colonne della matrice A ed agli elementi delle prime cinque righe e

cinque colonne della matrice B. Si noti che l'istruzione MAT INPUT assegna alla matrice A dimensioni attuali 3*4 e che l'istruzione 70 assegna alla matrice B dimensioni attuali 5*5.

```
LIST
FILE


0010 DISP "INTRODUCI I VALORI PER A            ";
0020 MAT INPUT A(3,4)
0030 PRINT
0040 PRINT "I VALORI DI A SONO:"
0050 MAT PRINT A;
0060 MAT A=ZER
0070 MAT B=ZER(5,5)
0080 PRINT
0090 PRINT "ORA I VALORI DI A SONO:"
0100 PRINT
0110 MAT PRINT A;
0120 PRINT
0130 PRINT
0140 PRINT "I VALORI DI B SONO:"
0150 MAT PRINT B;
0160 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
INTRODUCI I VALORI PER A            ?
1,2,3,4,5,6,7,8,9,0,9,8

I VALORI DI A SONO:
 1  2  3  4
 5  6  7  8
 9  0  9  8

ORA I VALORI DI A SONO:

 0  0  0  0
 0  0  0  0
 0  0  0  0


I VALORI DI B SONO:
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
```

Istruzione MAT INPUT

Funzione

Assegna agli elementi di una variabile multipla i dati introdotti da tastiera.

Formato

MAT INPUT array [(num-exp, num-exp)]

dove:

array
indica il nome di una variabile multipla, numerica o stringa, alla quale sono assegnati i valori introdotti da tastiera

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, rappresenta la nuova dimensione attuale della variabile multipla specificata con array.

Azione

L'esecuzione del programma è interrotta. Sul display un punto interrogativo (?) avverte l'utente di digitare n*m dati, dove m ed n sono i valori corrispondenti alle due espressioni specificate tra parentesi oppure le dimensioni attuali della matrice. I dati digitati sono assegnati nell'ordine, riga per riga, alla variabile multipla indicata nell'istruzione.

Note

1. Ogni dato digitato deve essere separato dai successivi mediante una virgola.

2. Ogni introduzione di dati deve essere completata con la pressione del tasto EOL. Se non sono stati digitati un numero di dati sufficienti ad esaurire la richiesta sul display due punti interrogativi indicano che il sistema è in attesa di altri dati da tastiera.

3. Quando il numero di dati digitati è pari al numero di quelli richiesti dal sistema, l'esecuzione del programma riprende.

4. Se sono digitati più valori di quelli richiesti dall'istruzione, il sistema visualizza il messaggio TOO MUCH DATA - EXCESS IGNORED e riprende l'esecuzione del programma.

5. Se è presente la parte opzionale, la matrice specificata nella istruzione assume come dimensioni attuali i valori corrispondenti alle due espressioni numeriche.

6. E' utile far precedere l'istruzione MAT INPUT da una istruzione DISP o PRINT che specifichino all'utente quali dati deve introdurre.

7. Ogni dato introdotto da tastiera deve essere dello stesso tipo (numerico o stringa) della variabile multipla a cui il valore è assegnato (stringa o numerica).

8. Se si introducono da tastiera stringhe con spazi iniziali e finali e/o virgole, le stringhe devono essere comprese tra virgolette.

9. Il prodotto delle dimensioni di allocazione della variabile multipla specificata nell'istruzione deve essere maggiore od uguale al prodotto dei valori delle due espressioni numeriche che definiscono le nuove dimensioni attuali.

10. Per ulteriori informazioni si vedano le note relative all'istruzione INPUT.

Esempi

1. La routine sottostante mostra l'impiego della istruzione MAT INPUT. Con l'istruzione 20 si assegnano agli elementi allocati in memoria principale per la matrice A (10 x 10) i valori numerici introdotti da tastiera. Si noti come, dopo ogni introduzione che non esaurisce la richiesta di dati da parte dell'istruzione, è visualizzato un doppio punto interrogativo (??). Con l'istruzione 70 si assegnano ai primi tre elementi delle prime due righe della variabile multipla stringa A$ le stringhe di caratteri *OLIVETTI P6066*.

```
LIST
FILE    *MAT15

0010 DISP "Introduci i valori per A        ";
0020 MAT INPUT A
0030 PRINT
0040 PRINT "I valori di A sono:"
0050 MAT PRINT A;
0060 DISP "Introduci i valori per A$        ";
0070 MAT INPUT A$(2,3)
0080 PRINT
0090 PRINT "I valori di A$ sono:"
0100 MAT PRINT A$
0110 END

END OF LISTING


RUN
Introduci i valori per A        ?
0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9
Introduci i valori per A        ??
0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9
Introduci i valori per A        ??
0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9

I valori di A sono:
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
 0  1  2  3  4  5  6  7  8  9
Introduci i valori per A$        ?
*Olivetti P6066*,*Olivetti P6066*,*Olivetti P6066*,*Olivetti P6066*
Introduci i valori per A$        ??
*Olivetti P6066*,*Olivetti P6066*

I valori di A$ sono:
*Olivetti P6066*                *Olivetti P6066*                *Olivetti P6066*
*Olivetti P6066*                *Olivetti P6066*                *Olivetti P6066*
```

2. La routine sottostante dichiara, istruzione 10, per gli elementi della variabile multipla stringa A$ una dimensione di allocazione di 30 caratteri. Con l'istruzione 30 vengono assegnati ai primi due elementi della prima riga di A$ le due stringhe di caratteri successivamente stampate con l'istruzione 60.

```
LIST
FILE    +MAT16

0010 DCL 30 A$()
0020 DISP "Introduci i valori per A$          ";
0030 MAT INPUT A$(1,2)
0040 PRINT
0050 PRINT "I valori di A$ sono:"
0060 MAT PRINT A$
0070 END

END OF LISTING


RUN
Introduci i valori per A$          ?
Il "Giorno" e` un quotidiano,!!!!!!!!!"""""""""$$$$$$$$

I valori di A$ sono:
Il "Giorno" e` un quotidiano    !!!!!!!!!"""""""""$$$$$$$$
```

Istruzione MAT PRINT

Funzione

Stampa i valori degli elementi di una o più variabili multiple nel formato standard.

Formato

MAT PRINT array [{,;} **array**] ... [,;]

dove:

array

è il nome di una variabile multipla, numerica o stringa, i cui valori sono stampati secondo un formato standard, sul tabulato della stampante integrata.

Azione

I valori degli elementi contenuti nelle variabili multiple sono convertiti nel formato specificato nella istruzione PRINT e stampati da sinistra a destra nell'ordine con cui sono presenti in ogni riga della variabile multipla.

La posizione dei caratteri nella linea di stampa è controllata da "," e ";" nel modo specificato nel paragrafo "Controllo della posizione dei caratteri nell'ambito della linea di stampa".

Le variabili multiple sono stampate con riferimento alle dimensioni attuali.

Controllo della posizione dei caratteri nell'ambito della linea di stampa

Il primo elemento di ogni riga di una variabile multipla è stampato nella prima posizione di una nuova linea di stampa.

In una istruzione MAT PRINT con più di una variabile mul ipla come operando, le variabili multiple devono essere separate da "," o ";".

Se una variabile multipla è seguita da "," il contenuto degli elementi di ogni riga della variabile multipla è stampato partendo dall'inizio di una delle 5 zone di stampa in cui è diviso il tabulato, come indicato in figura 5-2 (vedi istruzione PRINT).

Se una variabile multipla è seguita da ";" il contenuto degli elementi di ogni riga della variabile multipla è stampato di seguito al precedente nell'ambito della stessa linea di stampa.

Se dopo l'ultima variabile multipla di una istruzione MAT PRINT non vi è nè "," nè ";" la stampa del contenuto dei suoi elementi avviene come nel caso in cui vi sia ",".

Esempi

1. Nel programma sottostante l'istruzione 110 stampa i valori delle matrici A, B e C come si vede dopo l'intestazione "I valori di A, B e C, stampati usando ";" come separatore sono: ". Le istruzioni 160 e 170 producono lo stesso risultato. L'istruzione 220 stampa gli stessi valori ma secondo un altro formato, come si può vedere dopo la terza intestazione. Infine le istruzioni 280 e 290 producono lo stesso risultato.

```
LIST
FILE


0010 DISP "INTRODUCI I VALORI PER A        ";
0020 MAT INPUT A(4,4)
0030 DISP "INTRODUCI I VALORI DI B                  ";
0040 MAT INPUT B(3,3)
0050 DISP "INTRODUCI I VALORI DI C               ";
0060 MAT INPUT C(5,4)
0070 PRINT
0080 PRINT
0090 PRINT
0100 PRINT "I VALORI DI A,B e C, STAMPATI USANDO ; COME SEPARATORE SONO:"
0110 MAT PRINT A;B;C
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "CON DUE ISTRUZIONI PRODUCO LA STAMPA DI PRIMA."
0160 MAT PRINT A;
0170 MAT PRINT B;C
0180 PRINT
0190 PRINT
0200 PRINT
0210 PRINT "STAMPO I VALORI DI A,B e C USANDO , COME SEPARATORE."
0220 MAT PRINT A,B,C
0230 PRINT
0240 PRINT
0250 PRINT
0260 PRINT "CON DUE ISTRUZIONI PRODUCO LA STAMPA DI PRIMA."
0270 MAT PRINT A,
0280 MAT PRINT B,C
0290 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
INTRODUCI I VALORI PER A        ?
1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7
INTRODUCI I VALORI DI B                 ?
10,20,30,40,50,60,70,80,90
INTRODUCI I VALORI DI C               ?
-1,-2,-3,-4,-5,-6,-7,-8,-9,-1,-2,-3,-4,-5,-6,-7,-8,-9,-1,-2


I VALORI DI A,B e C, STAMPATI USANDO ; COME SEPARATORE SONO:
 1  2  3  4
 5  6  7  8
 9  1  2  3
 4  5  6  7
 10  20  30
 40  50  60
 70  80  90
-1              -2              -3              -4
-5              -6              -7              -8
-9              -1              -2              -3
-4              -5              -6              -7
-8              -9              -1              -2
```

```
CON DUE ISTRUZIONI PRODUCO LA STAMPA DI PRIMA.
 1  2  3  4
 5  6  7  8
 9  1  2  3
 4  5  6  7
 10  20  30
 40  50  60
 70  80  90
-1                -2                -3                -4
-5                -6                -7                -8
-9                -1                -2                -3
-4                -5                -6                -7
-8                -9                -1                -2


STAMPO I VALORI DI A,B e C USANDO , COME SEPARATORE.
 1                 2                 3                 4
 5                 6                 7                 8
 9                 1                 2                 3
 4                 5                 6                 7
 10                20                30
 40                50                60
 70                80                90
-1                -2                -3                -4
-5                -6                -7                -8
-9                -1                -2                -3
-4                -5                -6                -7
-8                -9                -1                -2


CON DUE ISTRUZIONI PRODUCO LA STAMPA DI PRIMA.
 1                 2                 3                 4
 5                 6                 7                 8
 9                 1                 2                 3
 4                 5                 6                 7
 10                20                30
 40                50                60
 70                80                90
-1                -2                -3                -4
-5                -6                -7                -8
-9                -1                -2                -3
-4                -5                -6                -7
-8                -9                -1                -2
```

```
Con due istruzioni produco la stampa di prima
 1              2              3              4              5
 6              7              8
 9              0              1              2              3
 4              5              6
 7              8              9              0              1
 2              3              4
 5              6              7              8              9
 0              1              2
 3              4              5              6              7
 8              9              0
 1              2              3              4              5
 6              7              8
 9              0              1              2              3
 4              5              6
 7              8              9              0              1
 2              3              4
 10             20             30             40             50
 60
 70             80             90             10             20
 30
 40             50             60             70             80
 90
 10             20             30             40             50
 60
 70             80             90             10             20
 30
 40             50             60             70             80
 90
-1             -2             -3             -4
-5             -6             -7             -8
-9             -1             -2             -3
-4             -5             -6             -7
-8             -9             -1             -2
```

2. La routine sottostante, dopo aver assegnato le stringhe introdotte da tastiera alle variabile multiple stringa A$ e B$, le stampa con due formati diversi secondo quanto specificato nelle istruzioni 90 e 150.

```
LIST
FILE


0010 DISP "Introduci i valori per A$                          ";
0020 MAT INPUT A$(2,3)
0030 DISP "Introduci i valori per B$                           ";
0040 MAT INPUT B$(2,2)
0050 PRINT
0060 PRINT
0070 PRINT
0080 PRINT "Stampo i valori di A$ e B$ usando ;"
0090 MAT PRINT A$;B$;
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "Stampo i valori di A$ e B$ usando ,"
0140 MAT PRINT A$,B$
0150 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per A$                          ?
*BASIC P6066*,*BASIC P6066*,*BASIC P6066*,*BASIC P6066*,*BASIC P6066*
Introduci i valori per A$                          ??
*BASIC P6066*
Introduci i valori per B$                           ?
*Olivetti P6066*,*Olivetti P6066*,*Olivetti P6066*,*Olivetti P6066*


Stampo i valori di A$ e B$ usando ;
*BASIC P6066**BASIC P6066**BASIC P6066*
*BASIC P6066**BASIC P6066**BASIC P6066*
*Olivetti P6066**Olivetti P6066*
*Olivetti P6066**Olivetti P6066*


Stampo i valori di A$ e B$ usando ,
*BASIC P6066*   *BASIC P6066*   *BASIC P6066*
*BASIC P6066*   *BASIC P6066*   *BASIC P6066*
*Olivetti P6066*                *Olivetti P6066*
*Olivetti P6066*                *Olivetti P6066*
```

Istruzione MAT PRINT USING

Funzione

Stampa i valori degli elementi di una o più variabili multiple in un formato predefinito in una istruzione immagine.

Formato

MAT PRINT USING {**line-num** | **string-var**}, **array** [, **array**] ...

dove:

line-num
indica il numero di linea di una istruzione IMMAGINE

string-var
indica una variabile stringa, semplice o con indice, il cui contenuto rappresenta una immagine di formato

array
indica il nome di una variabile multipla, numerica o stringa, i cui valori sono stampati secondo un formato definito dall'utente.

Azione

I valori degli elementi di ogni riga delle matrici specificate nella istruzione MAT PRINT sono convertiti nel formato specificato nella istruzione IMMAGINE il cui numero di linea è specificato con line-num o dal contenuto della variabile stringa, specificata con string-var, e stampati, da sinistra a destra, nell'ordine con sui sono presenti in ogni riga della variabile multipla.

L'associazione tra valori da stampare e campi della immagine di formato è data, da sinistra a destra, nell'ordine con cui i valori compaiono negli elementi della riga della matrice ed "i campi immagine" nella immagine di formato.

Note

1. Ogni istruzione PRINT USING stampa i valori degli elementi di ogni riga delle variabili multiple a partire da una nuova riga di stampa.

2. Se vi sono più elementi nella riga di una variabile multipla che campi di formato, nella immagine di formato, gli elementi in più sono stampati sulle righe di stampa successive con lo stesso formato.

3. Se vi sono più campi di formato nella immagine di formato che elementi in una riga di una variabile multipla, in corrispondenza dei campi immagine eccedenti vengono generati degli spazi.

4. I valori degli elementi di una variabile multipla ed i campi di formato, nella immagine di formato, devono essere coerenti: ad una matrice numerica deve corrispondere una immagine di formato di tipo numerico; ad una variabile multipla stringa deve corrispondere una immagine di formato di tipo stringa.

Esempi

1. Nella routine sottostante l'istruzione 90 stampa i valori delle matrici A e B secondo il formato definito con l'istruzione 50. L'istruzione 150 stampa gli stessi valori con il formato definito dal contenuto della variabile I$. Poichè il formato è composto da due soli campi, il terzo valore di ogni riga delle matrici A e B è stampato su di una nuova riga di stampa. Infine l'istruzione 210 stampa gli stessi valori con il formato definito dall'istruzione 160. In questo caso, il campo immagine in più è ignorato.

```
LIST
FILE    +MAT40

0010 DISP "Introduci i valori per A       ";
0020 MAT INPUT A(3,3)
0030 DISP "Itroduci i valori per B       ";
0040 MAT INPUT B(3,3)
0050 :    ###           ###           ###
0060 PRINT
0070 PRINT
0080 PRINT "Stampo i valori con l'immagine definita nell'istruzione 50"
0090 MAT PRINT USING 50,A,B
0100 LET I$="###          ###"
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "Stampo i valori di A e B con l'immagine contenuta in I$."
0150 MAT PRINT USING I$,A,B
0160 :    ####      ####      ####      ####
0170 PRINT
0180 PRINT
0190 PRINT
0200 PRINT "Stampo i valori con l'immagine definita nell'istruzione 160."
0210 MAT PRINT USING 160,A,B
0220 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per A      ?
1,2,3,4,5,6,7,8,9
Itroduci i valori per B      ?
-1,-2,-3,-4,-5,-6,-7,-8,-9


Stampo i valori con l'immagine definita nell'istruzione 50
      1             2             3
      4             5             6
      7             8             9
     -1            -2            -3
     -4            -5            -6
     -7            -8            -9


Stampo i valori di A e B con l'immagine contenuta in I$.
  1            2
  3
  4            5
  6
  7            8
  9
 -1           -2
 -3
 -4           -5
 -6
 -7           -8
 -9


Stampo i valori con l'immagine definita nell'istruzione 160.
       1         2         3
       4         5         6
       7         8         9
      -1        -2        -3
      -4        -5        -6
      -7        -8        -9
```

2. Nella routine sottostante l'istruzione 90 stampa le stringhe assegnate da tastiera ad A$ e B$ con il formato definito nell'istruzione 50. L'istruzione 150 stampa le stesse stringhe con il formato definito dal contenuto di I$. Poichè in questo caso il formato è composto da due soli campi immagine, il terzo valore di ogni riga delle matrici viene stampato su di una nuova riga di stampa. Infine l'istruzione 210 stampa gli stessi valori con il formato definito dall'istruzione 160. In questo caso il formato ha un campo immagine in più del numero di elementi di ogni riga delle variabili multiple stringa per cui tale campo è ignorato.

```
LIST
FILE    +MAT19

0005 DCL 80I$
0010 DISP "Introduci i valori per A$              ";
0020 MAT INPUT A$(3,3)
0030 DISP "Introduci i valori per B$          ";
0040 MAT INPUT B$(3,3)
0050     'LLLLLLLLLLLLLLLL     'CCCCCCCCCCCCCCCC      'RRRRRRRRRRRRRRRR
0060 PRINT
0070 PRINT
0080 PRINT "Stampo i valori con l'immagine definita nell'istruzione 50"
0090 MAT PRINT USING 50,A$,B$
0100 LET I$="'LLLLLLLLLLLLLLLL          'LLLLLLLLLLLLLLLL"
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "Stampo i valori di A$ e B$ con l'immagine contenuta in I$."
0150 MAT PRINT USING I$,A$,B$
0160  'LLLLLLLLLLLLLLLL  'LLLLLLLLLLLLLLLL  'LLLLLLLLLLLLLLLL  'LLLLLLLLLLLLLLLL
0170 PRINT
0180 PRINT
0190 PRINT
0200 PRINT "Stampo i valori  con l'immagine definita nell'istruzione 160."
0210 MAT PRINT USING 160,A$,B$
0220 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Introduci i valori per A$              ?
AAAAAAAAAAAAAAAA,BBBBBBBBBBBBBBBB,CCCCCCCCCCCCCCCC,DDDDDDDDDDDDDDDD
Introduci i valori per A$              ??
EEEEEEEEEEEEEEEE,FFFFFFFFFFFFFFFF,GGGGGGGGGGGGGGGG,HHHHHHHHHHHHHHHH
Introduci i valori per A$              ??
IIIIIIIIIIIIIIII
Introduci i valori per B$          ?
a,b,c,d,e,f,g,h,i


Stampo i valori con l'immagine definita nell'istruzione 50
   AAAAAAAAAAAAAAAA      BBBBBBBBBBBBBBBB        CCCCCCCCCCCCCCCC
   DDDDDDDDDDDDDDDD      EEEEEEEEEEEEEEEE        FFFFFFFFFFFFFFFF
   GGGGGGGGGGGGGGGG      HHHHHHHHHHHHHHHH        IIIIIIIIIIIIIIII
   a                          b                        c
   d                          e                        f
   g                          h                        i
```

```
Stampo i valori di A$ e B$ con l immagine contenuta in I$.
AAAAAAAAAAAAAAAA            BBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDD            EEEEEEEEEEEEEEEE
FFFFFFFFFFFFFFFF
GGGGGGGGGGGGGGGG            HHHHHHHHHHHHHHHH
IIIIIIIIIIIIIIII
a                           b
c
d                           e
f
g                           h
i


Stampo i valori  con l immagine definita nell'istruzione 160.
AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBB   CCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDD   EEEEEEEEEEEEEEEE   FFFFFFFFFFFFFFFF
GGGGGGGGGGGGGGGG   HHHHHHHHHHHHHHHH   IIIIIIIIIIIIIIII
a                  b                  c
d                  e                  f
g                  h                  i
```

Istruzione MAT READ

Funzione

Assegna agli elementi di una o più variabili multiple i dati contenuti nel file interno definito con le istruzioni DATA.

Formato

MAT READ array [(num-exp, num-exp)] [, array [(num-exp, num-exp)]] ...

dove:

array
indica il nome di una variabile multipla, numerica o stringa, ai cui elementi sono assegnati i valori dal file dati interno

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo , rappresenta la nuova dimensione attuale della variabile multipla.

Azione

Agli elementi delle variabili multiple specificate nella istruzione sono assegnati ordinatamente, riga per riga, e variabile per variabile, i valori del file dati interno iniziando dalla posizione indicata dal pointer (vedi istruzione DATA).

Man mano che ogni valore è assegnato ad ogni elemento della variabile multipla, il pointer indica la posizione del dato successivo del file dati interno.

Se sono specificate le espressioni numeriche della parte opzionale, queste ultime sono eseguite ed i valori ottenuti, arrotondati all'intero più prossimo, rappresentano le nuove dimensioni attuali della variabile multipla.

Se non sono specificate le espressioni numeriche suddette, i valori del file dati interno sono assegnati agli elementi della variabile multipla secondo le sue dimensioni attuali.

Note

1. Ogni elemento di una variabile multipla presente in una istruzione MAT READ deve avere un valore associato nel file dati interno, altrimenti l'esecuzione del programma è sospesa e viene visualizzato un messaggio di errore non recuperabile. Premendo BREAK il sistema commuta nello stato comandi.

2. Agli elementi delle variabili multiple di tipo stringa possono essere assegnati dati numerici che sono considerati stringhe di caratteri.

3. Ad un elemento di una matrice numerica non deve essere assegnata una stringa.

4. Se ad un elemento di una matrice dichiarata in singola precisione è assegnato un dato numerico in virgola mobile con più di 6 cifre significative, ma con esponente che rientri nel range della singola precisione, la mantissa viene troncata a 6 cifre significative.

5. Se ad un elemento di una matrice dichiarata in singola precisione è assegnato un dato numerico in virgola mobile con esponente nella zona di OVERFLOW per la singola precisione, il sistema visualizza un messaggio di errore recuperabile e commuta nello stato di debugging, assegnando all'elemento suddetto il valore 9.99999E63 oppure -9.99999E63. Premendo CONTINUE l'elaborazione del programma prosegue; premendo BREAK l'elaborazione termina.

6. Se ad un elemento di una matrice dichiarata in singola precisione è assegnato un dato numerico in virgola mobile con esponente nella zona di UNDERFLOW per il tipo di precisione specificato, il sistema visualizza un messaggio di errore recuperabile e commuta nello stato di debugging, assegnando all'elemento suddetto il valore zero. Premendo CONTINUE l'elaborazione del programma prosegue; premendo BREAK l'elaborazione termina.

7. Se ad un elemento di una variabile multipla di tipo stringa viene assegnata una stringa con più caratteri della lunghezza di allocazione dichiarata per gli elementi della variabile multipla, il sistema visualizza un messaggio di errore recuperabile e commuta nello stato di debugging, assegnando all'elemento suddetto la stringa troncata sulla destra

dei caratteri eccedenti. Premendo CONTINUE l'elaborazione del programma prosegue; premendo BREAK l'elaborazione termina.

Esempio

Nella routine sottostante tutte le variabili numeriche sono rappresentate in singola precisione e tutti gli elementi di A$ possono avere al massimo 30 caratteri. Le istruzioni 30, 40 e 50 definiscono un file dati interno da cui sono prelevati i valori mediante le istruzioni MAT READ. L'istruzione 60 assegna ai primi due elementi delle prime due righe della matrice A i primi quattro valori del file dati interno suddetto. Si noti come per il primo ed ultimo valore la mantissa è stata troncata dopo le prime 6 cifre significative, perchè la matrice ha elementi in singola precisone. I successivi quattro valori del file dati interno sono assegnati agli elementi di A$ con l'istruzione 160. L'istruzione 180 ripone il pointer del file dati interno all'inizio del file. L'istruzione 220, infine, assegna ai primi due elementi delle prime due righe di A$ (le dimensioni attuali sono state definite con la precedente istruzione 160) i primi 4 valori contenuti nel file dati interno, come si vede dalla stampa prodotta con l'istruzione 240.

```
FILE     RMATRD

0010 DCL SINGLE
0020 DCL 30(A$())
0030 DATA 123456789E34,123456E34,-123456E-34,-123456789E-34
0040 DATA * Olivetti P6066*,"AREA,VOLUME","   PESO   "
0050 DATA Temperatura "media" a Parigi
0060 MAT READ A(2,2)
0070 PRINT
0080 PRINT
0090 PRINT
0100 PRINT "I valori di A sono:"
0110 MAT PRINT A;
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "I valori di A$ sono:"
0160 MAT READ A$(2,2)
0170 MAT PRINT A$;
0180 RESTORE
0190 PRINT
0200 PRINT
0210 PRINT
0220 MAT READ A$
0230 PRINT "Ora i valori di A$ sono:"
0240 MAT PRINT A$
0250 END

END OF LISTING
```

```
I valori di A sono:
 1.2345600E+42  1.2345600E+39
-1.2345600E-29 -1.2345600E-26


I valori di A$ sono:
* Olivetti P6066*AREA,VOLUME
   PESO   Temperatura "media" a Parigi


Ora i valori di A$ sono:
123456789E34     123456E34
-123456E-34      -123456789E-34
```

Istruzione MAT READ:

Funzione

Assegna agli elementi di una o più variabili multiple i dati contenuti in un file dati esterno.

Formato

MAT READ: file-designator, array [(num-exp, num-exp)] [, array [(num-exp, num-exp)]] ... [EOF line-num]

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, rappresenta il designatore di un file dati esterno

array
indica il nome di una variabile multipla numerica o stringa

num-exp
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, rappresenta la nuova dimensione attuale della variabile multipla

line-num
è il numero di linea di una istruzione del programma.

Azione

L'espressione numerica relativa a file designator è eseguita ed il valore ottenuto, arrotondato all'intero più prossimo nd, specifica il file dati esterno da cui sono prelevati i valori da assegnare agli elementi delle variabili multiple indicate nella istruzione.

I valori da assegnare sono prelevati dal file con numero designatore nd, iniziando dal dato su cui è posizionato il pointer del file, ed assegnati nell'ordine, riga per riga, variabile multipla per variabile multipla, agli elementi delle variabili multiple presenti nella istruzione.

Il pointer del file si sposta man mano e si posiziona dopo l'ultimo dato letto.

Se una variabile multipla è seguita dalla parte opzionale, le espressioni numeriche sono eseguite ed i valori ottenuti, arrotondati all'intero più prossimo m ed n, specificano quali sono le nuove dimensioni attuali della matrice. Quindi i valori del file sono assegnati ad una matrice di dimensioni attuali m*n.

Se dopo una variabile multipla non si specifica la parte opzionale, i valori del file dati esterno sono assegnati alla variabile multipla suddetta secondo le sue dimensioni attuali.

Se il pointer del file è posizionato dopo l'ultimo dato registrato in un file sequenziale, l'esecuzione dell'istruzione MAT READ: dà una segnalazione di errore e l'esecuzione del programma è sospesa; se, però, è presente l'opzione EOF, il controllo della esecuzione del programma passa alla istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Se il pointer di un file ad accesso diretto è posizionato dopo l'ultima parola allocata per il file (vedi comando CREATE), l'esecuzione della istruzione MAT READ: dà una segnalazione di errore; se, però, è presente l'opzione EOF, il controllo della esecuzione del programma passa alla istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Note

1. Se il file è stato dichiarato, con il comando CREATE, ad accesso diretto, si possono leggere i dati che si vogliono. Infatti l'istruzione SETW: posiziona il pointer del file su un dato qualunque del file esterno.

2. Se il file è stato dichiarato, con il comando CREATE, ad accesso sequenziale, l'istruzione MAT READ: deve essere preceduta da una delle seguenti istruzioni riferite allo stesso file: RESTORE:, READ:, MAT READ:.

3. I valori assegnati agli elementi delle variabili multiple devono essere dello stesso tipo di queste

ultime (numeriche o stringa).

4. Il risultato della espressione numerica, arrotondato all'intero più prossimo, che determina il numero designatore del file esterno da cui sono letti i valori deve essere maggiore di zero e minore od uguale al numero di file accessibili contemporaneamente dal programma (dichiarato con l'istruzione FILES).

5. Il file esterno deve essere stato "aperto" mediante una precedente istruzione FILES o FILE:.

6. Per ulteriori osservazioni si vedano le note contenute nella descrizione dell'istruzione READ:.

Esempi

1. La routine sottostante è eseguita dopo aver eseguito la routine riportata nell'esempio 1 della descrizione della istruzione WRITE:. L'istruzione 10 assegna ai file sequenziali SEQ1 e SEQ2 i numeri designatori 1 e 2. L'istruzione 20 pone il file SEQ1 in lettura ed il suo pointer all'inizio del file. L'istruzione 30 legge i dati del file esterno SEQ1 e li assegna ai primi tre elementi delle prime tre righe di A ed ai primi tre elementi delle prime tre righe di B. L'istruzione 80 stampa i valori assegnati ad A da file dati esterno. L'istruzione 130 stampa i valori assegnati a B da file dati esterno. L'istruzione 140 pone il pointer del file SEQ2 all'inizio del file e ne permette la lettura. L'istruzione 150 legge le stringhe di dati del file SEQ2 e le assegna ai primi due elementi delle prime due righe della variabile multipla A$ ed ai primi tre elementi delle prime due righe di B$. Le istruzioni 190 e 240 stampano i valori assegnati ad A$ e B$.

```
LIST
FILE     MATRE1

0010 FILES SEQ1,SEQ2
0020 RESTORE :1
0030 MAT READ :1,A(3,3),B(2,3)
0040 PRINT
0050 PRINT
0060 PRINT
```

```
0070 PRINT "I valori di A() sono:"
0080 MAT PRINT A;
0090 PRINT
0100 PRINT
0110 PRINT
0120 PRINT "I valori di B() sono:"
0130 MAT PRINT B
0140 RESTORE :2
0150 MAT READ :2,A$(2,2),B$(2,3)
0160 PRINT
0170 PRINT
0180 PRINT
0190 PRINT "I valori di A$ sono:"
0200 MAT PRINT A$
0210 PRINT
0220 PRINT
0230 PRINT
0240 PRINT "I valori di B$ sono:"
0250 MAT PRINT B$
0255 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****


I valori di A() sono:
 1  2  3
 4  5  6
 7  8  9


I valori di B() sono:
 2                 4                 6
 8                 16                25


I valori di A$ sono:
PRIMO DATO        SECONDO DATO
TERZO DATO        QUARTO DATO


I valori di B$ sono:
Biella            Milano            Napoli
Roma              Torino            Venezia
```

2. La routine sottostante è eseguita dopo aver eseguito la routine riportata nell'esempio 1 della descrizione dell'istruzione WRITE:. L'istruzione 10 assegna ai file esterni, ad accesso diretto, DIR1 e DIR2 i numeri designatori 1 e 2. L'istruzione 20 pone il pointer del file DIR1 all'inizio della decima parola. L'istruzione 30 legge dalla decima parola del file DIR1 i dati da assegnare ai primi tre elementi delle prime tre righe di A. L'istruzione 40 pone il pointer del file DIR1 all'inizio della 30-esima parola e quindi l'istruzione 50 assegna i dati che iniziano da tale parola ai primi due elementi delle prime due righe di B. L'istruzione 60 legge dall'inizio del file DIR2 i dati e

li assegna ordinatamente ai primi due elementi delle prime due righe della variabile A$ ed ai primi tre elementi delle prime due righe della variabile B$. Le istruzioni 110, 160, 210 e 260 stampano i valori assegnati alle matrici A e B ed alle variabili multiple stringa A$ e B$.

```
LIST
FILE    +MATRE2

0010 FILES DIR1;DIR2
0020 SETW :1 TO 10
0030 MAT READ :1,A(3,3)
0040 SETW :1 TO 30
0050 MAT READ :1,B(2,2)
0060 MAT READ :2,A$(2,2),B$(2,3)
0070 PRINT
0080 PRINT
0090 PRINT
0100 PRINT "I valori di A() sono:"
0110 MAT PRINT A;
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "I valori di B() sono:"
0160 MAT PRINT B;
0170 PRINT
0180 PRINT
0190 PRINT
0200 PRINT " I valori di A$() sono:"
0210 MAT PRINT A$
0220 PRINT
0230 PRINT
0240 PRINT
0250 PRINT "I valori di B$() sono:"
0260 MAT PRINT B$
0270 END

END OF LISTING


RUN


I valori di A() sono:
 1  2  3
 4  5  6
 7  8  9


I valori di B() sono:
 2  4
 6  8


 I valori di A$() sono:
PRIMO DATO      SECONDO DATO
TERZO DATO      QUARTO DATO


I valori di B$() sono:
Biella          Milano          Napoli
Roma            Torino          Venezia
```

Istruzione MAT WRITE:

Funzione

Registra in un file dati esterno i valori degli elementi di una o più variabili multiple specificate.

Formato

MAT WRITE: file-designator, array [, array] ... [EOF line-num]

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica il designatore del file dati esterno in cui devono essere registrati i valori degli elementi delle variabili multiple specificate

array
è il nome di una variabile multipla da cui sono prelevati i valori da registrare sul file dati esterno

line-num
indica il numero di linea di una istruzione del programma.

Azione

L'espressione riferita a file-designator è eseguita ed il valore ottenuto, arrotondato all'intero più prossimo nd, costituisce il numero designatore del file nel quale saranno registrati i valori degli elementi delle variabili multiple specificate nella istruzione.

I valori degli elementi delle variabili multiple indicate nella istruzione sono registrati nel file suddetto, iniziando dalla posizione indicata dal pointer del file. La registrazione avviene elemento dopo elemento, riga dopo riga, nell'ordine da sinistra a destra per ogni varriabile multipla indicata nell'istruzione. Se il pointer del file è posizionato dopo l'ultima parola allocata per il file (sequenziale o ad accesso diretto)

nella libreria, l'esecuzione dell'istruzione MAT WRITE: dà una segnalazione di errore non recuperabile; se, però, è presente l'opzione EOF, il controllo della esecuzione del programma passa alla istruzione il cui numero di linea è specificato nella opzione stessa con line-num e non vi è alcuna segnalazione di errore. In ogni caso i valori per i quali esiste sufficiente spazio sono registrati nel file.

Note

1. Se il file è sequenziale, l'istruzione MAT WRITE: deve essere preceduta da una istruzione SCRATCH, APPEND:, WRITE:, o MAT WRITE:.

2. Se il file è ad accesso casuale e si vogliono registrare i dati iniziando da una parola specifica del file, l'istruzione MAT WRITE: deve essere preceduta da una istruzione SETW:.

3. Se l'istruzione MAT WRITE: è eseguita dopo una istruzione FILES, FILE:, SCRATCH: o RESTORE: la registrazione del file inizia dalla prima parola del file stesso (l'istruzione SCRATCH: è usata solo per file sequenziali).

4. nd deve essere maggiore di zero e minore o uguale al numero di file accessibili contemporaneamente dal programma (dichiarato con l'istruzione FILES).

5. Per ulteriori informazioni si vedano le note riportate nella descrizione della istruzione WRITE:.

Esempi

1. Nella routine sottostante l'istruzione 10 assegna ai file SEQ1 e SEQ2 (sequenziali) e DIR1 e DIR2 (ad accesso diretto) rispettivamente i numeri designatori 1,2,3 e 4. L'istruzione 100 pone il pointer del file SEQ1 nella prima posizione del file e permette di registrare in esso i valori contenuti nelle matrici A e B (istruzione 110). L'istruzione 120 pone il pointer del file SEQ2 nella prima posizione del file e permette di registrare in esso i valori delle variabili multiple A$ e B$ con l'istruzione 130. L'istruzione 140 pone il pointer del file DIR1 all'inizio della decima parola. L'istruzione 150 registra dalla decima parola del file DIR1 i valori della matrice A. L'istruzione 160 pone il pointer del file DIR1 all'inizio della 30-esima parola da

dove l'istruzione 170 registra i valori della matrice B. Infine l'istruzione 180 registra dall'inizio del file DIR2 le stringhe contenute negli elementi delle variabili multiple A$ e B$.

```
LIST
FILE     MATWR1

0010 FILES SEQ1;SEQ2;DIR1;DIR2
0020 DISP "Introduci i valori per A()       ";
0030 MAT INPUT A(3,3)
0040 DISP "Introduci i valori per B()       ";
0050 MAT INPUT B(2,3)
0060 DISP "Introduci i valori per A$()      ";
0070 MAT INPUT A$(2,2)
0080 DISP "Introduci i valori per B$()      ";
0090 MAT INPUT B$(2,3)
0100 SCRATCH :1
0110 MAT WRITE :1,A,B
0120 SCRATCH :2
0130 MAT WRITE :2,A$,B$
0140 SETW :3 TO 10
0150 MAT WRITE :3,A
0160 SETW :3 TO 30
0170 MAT WRITE :3,B
0180 MAT WRITE :4,A$,B$
0190 END

END OF LISTING


RUN
Introduci i valori per A()        ?
1,2,3,4,5,6,7,8,9
Introduci i valori per B()       ?
2,4,6,8,16,25
Introduci i valori per A$()       ?
PRIMO DATO,SECONDO DATO,TERZO DATO,QUARTO DATO
Introduci i valori per B$()       ?
Biella,Milano,Napoli,Roma,Torino,Venezia
```

2. Nella routine sottostante l'iztruzione 10 assegna ai file ad accesso diretto DIRE1 e DIRE2 i numeri designatori 1 e 2. L'istruzione 40 pone il pointer del file DIRE1 all'inizio della decima parola del file da dove i valori della matrice A sono registrati con l'istruzione 50. L'istruzione 80 pone il pointer del file DIRE2 all'inizio della 20-esima parola da dove l'istruzione 90 registra i valori degli elementi della matrice B. L'istruzione 100 pone il pointer del file DIRE1 all'inizio della decima parola del file, da dove l'istruzione 110 legge i valori da assegnare ai primi 5 elementi delle prime 5 righe della matrice A. L'istruzione 170 pone il pointer del file DIRE2 all'inizio della 20-esima parola da dove l'istruzione 180 legge i valori da assegnare ai primi 5 elementi delle prime 5 ri-

ghe di B. L'istruzione 200 assegna a C il prodotto, righe per colonne, delle matrici A e B. L'istruzione 210 pone il pointer del file DIRE1 all'inizio della decima parola da dove l'istruzione 220 registra i valori della matrice C, aggiornando così il file dati. Infine l'istruzione 230 ripone il pointer del file DIRE1 all'inizio della decima parola da dove l'istruzione 240 legge i valori da assegnare nuovamente ai primi 5 elementi delle prime 5 righe di C.

```
LIST
FILE     MATRW

0010 FILES DIRE1;DIRE2
0020 DISP "Introduzione dei valori per A()  ";
0030 MAT INPUT A(5,5)
0040 SETW :1 TO 10
0050 MAT WRITE :1,A
0060 DISP "Introduci i valori per B()       ";
0070 MAT INPUT B(5,5)
0080 SETW :2 TO 20
0090 MAT WRITE :2,B
0100 SETW :1 TO 10
0110 MAT READ :1,A
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "I valori di A() sono:"
0160 MAT PRINT A;
0170 SETW :2 TO 20
0180 MAT READ :2,B(5,5)
0181 PRINT
0182 PRINT
0183 PRINT
0184 PRINT "I valori di B() sono:"
0190 MAT PRINT B;
0200 MAT C=A*B
0210 SETW :1 TO 10
0220 MAT WRITE :1,C
0230 SETW :1 TO 10
0240 MAT READ :1,C(5,5)
0250 PRINT
0260 PRINT
0270 PRINT
0280 PRINT "I valori di registrati nel file DIRE1 a partire dalla 10a parola :"
0290 MAT PRINT C
0300 END

END OF LISTING

RUN
**** FORMALLY CORRECT PROGRAM ****
Introduzione dei valori per A()  ?
1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7
Introduci i valori per B()       ?
1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7


I valori di A() sono:
 1  2  3  4  5
 6  7  8  9  1
 2  3  4  5  6
 7  8  9  1  2
 3  4  5  6  7
```

```
I valori di B() sono:
 1  2  3  4  5
 6  7  8  9  1
 2  3  4  5  6
 7  8  9  1  2
 3  4  5  6  7


I valori di registrati nel file DIRE1 a partire dalla 10a parola :
 62              77              92              71              68
 130             161             192             142             110
 81              101             121             96              89
 86              113             140             158             113
 100             125             150             121             110
```

6. STATO CALCOLI IMMEDIATI

Il P6066 può essere impiegato per eseguire istantaneamente calcoli complessi senza dover utilizzare un programma BASIC. In questo caso il sistema deve essere nello stato calcoli immediati. Per commutare il sistema dallo stato comandi allo stato calcoli immediati si deve premere il tasto di console CALC MODE. Il sistema rimane nello stato suddetto finchè non si preme di nuovo il tasto CALC MODE che ricommuta il sistema nello stato comandi.

Lo stato calcoli immediati è naturalmente utile per la sua ovvia funzione: il sistema può essere usato come macchina da calcolo. Tuttavia, esso ha una seconda e forse più importante funzione: poichè le operazioni eseguite nello stato calcoli immediati non cancellano il contenuto della memoria principale, questo stato può essere un valido strumento per verificare i risultati ottenuti da un programma.

Molti programmi che sono eseguiti senza produrre alcuna segnalazione di errore possono, tuttavia, fornire dei risultati errati a causa della loro struttura logica. Fornendo la possibilità di verificare immediatamente i risultati di un programma -- mentre il programma è ancora nella memoria principale -- lo stato calcoli immediati offre un mezzo estremamente rapido e conveniente per scoprire degli errori di programmazione che altrimenti sarebbe difficile rilevare.

Quando il sistema P6066 è nello stato calcoli immediati si può introdurre:

- un comando di sistema
- un comando di richiamo di un programma di utilità
- una espressione da eseguire immediatamente

I comandi di sistema sono introdotti ed eseguiti come spiegato nel capitolo 3. Dopo che è eseguito un comando, il sistema ricommuta automaticamente nello stato comandi.

I comandi di richiamo dei programmi di utilità sono introdotti ed eseguiti come spiegato nella appendice A. Dopo che il programma è eseguito, il sistema ricommuta automaticamente nello stato comandi.

Le espressioni tipiche dello stato calcoli immediati sono descritte nei paragrafi che seguono.

Introduzione ed esecuzione di espressioni nello stato calcoli immediati

Quando il P6066 è nello stato calcoli immediati, si può richiedere al sistema di:

- eseguire una espressione numerica
- assegnare agli argomenti delle funzioni trigonometriche una particolare unità di misura (radianti, gradi sessagesimali o centesimali)
- assegnare un contenuto predefinito ai tasti funzione

Espressioni numeriche nello stato calcoli immediati

Una espressione numerica è composta da un insieme di operandi uniti da uno o più operatori. La valutazione di una espressione numerica fornisce come risultato un numero. Un operando numerico può essere una:

- costante numerica
- variabile numerica
- funzione numerica di sistema
- funzione numerica definita dall'utente
- espressione numerica racchiusa tra parentesi

Costanti numeriche

Una costante numerica, nel sistema P6066, è un numero espresso nel sistema decimale. Come si può vedere negli esempi seguenti, una costante numerica può essere introdotta, visualizzata o stampata con tre formati diversi: intero, virgola fissa o virgola mobile.

Formato intero	In virgola fissa	In virgola mobile
5	5.0	5.0 E +0
-5	-5.0	-5.0 E +0
+5	+5.0	+5.0 E +0

Formato intero

Una costante numerica espressa nel formato intero è composta da una o più cifre precedute, opzionalmente, dal segno. Da tastiera si possono introdurre costanti numeriche intere con al massimo 13 cifre. Il massimo numero che si può introdurre è : 9999999999999.

Formato in virgola fissa

Una costante numerica espressa nel formato in virgola fissa è composta da una o più cifre separate dal punto decimale e precedute, opzionalmente, dal segno. Una costante numerica nel formato in virgola fissa può essere introdotta da tastiera con non più di 13 cifre.

Nota: La costante π è una costante interna del sistema. Al contrario delle altre costanti numeriche essa è richiamata da tastiera mediante il nome PI. Il suo valore è 3.141592653590.

Formato in virgola mobile

Una costante numerica nel formato in virgola mobile (noto anche come "notazione scientifica") consiste in un numero intero od in virgola fissa seguito da E e da una o due cifre, opzionalmente precedute dal segno. Il massimo numero che si può introdurre da tastiera è: 9.999999999999E+99.

Rappresentazione interna

La rappresentazione interna di un numero è la forma con cui è rappresentato in memoria principale. Tutti i numeri sono rappresentati in memoria principale con due parti: la mantissa e l'esponente. Per esempio 5665 è rappresentato in memoria principale come: 5.665000000000E+03 di cui 5.665000000000 è la mantissa e 03 è l'esponente. La mantissa viene normalizzata in modo che essa sia sempre composta da una cifra compresa tra 1 e 9, seguita da 12 cifre decimali. Il valore rappresentato in memoria principale è dato dal prodotto della mantissa per 10 elevato all'esponente indicato.

Campo della rappresentazione interna

I numeri che possono essere elaborati devono rientrare nel campo di definizione da -9.999999999999E+99 a -1E-99, lo zero, da +1E-99 a +9.999999999999E+99, come indicato nella figura 6-1. La parte tratteggiata indica i numeri non definiti nella rappresentazione interna.

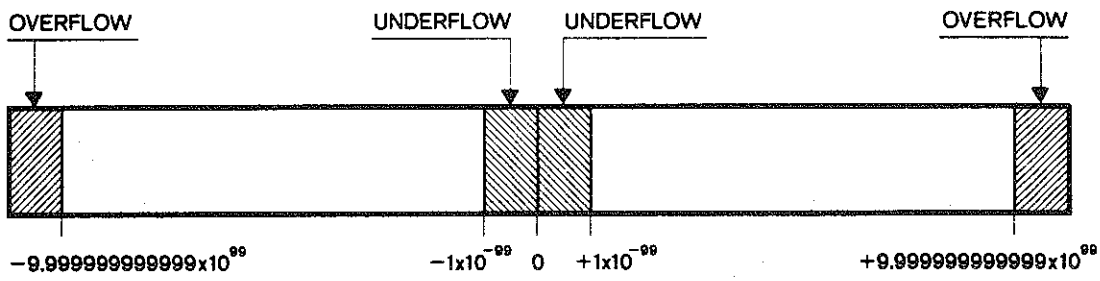


Figura 6-1 Campo della rappresentazione interna dei numeri

I numeri minori di 9-999999999999E99 e maggiori di 9.999999999999E99 appartengono alle zone di OVERFLOW, ossia esprimono grandezze in valore assoluto più grandi di quelle rappresentabili in memoria principale.

I numeri maggiori di $-1*10^{-99}$ed inferiori a $+1*10^{-99}$, ma diversi da zero, appartengono alle zone di UNDERFLOW, ossia esprimono grandezze in valore assoluto più piccole di quelle rappresentabili in memoria principale.

Se durante l'elaborazione di una espressione si ottiene un risultato parziale con valore compreso in una zona di OVERFLOW, l'esecuzione della espressione è portata a termine assumendo come valore intermedio -9.999999999999E99 oppure 9.999999999999E99. Il risultato finale è stampato, se il tasto di console PRINT ALL è illuminato, ed il sistema visualizza il messaggio di errore ERROR 3.

Se durante l'elaborazione di una espressione si ottiene un risultato parziale con valore compreso in una zona di UNDERFLOW, l'esecuzione della espressione è portata a termine assumendo come valore intermedio zero. Il risultato finale è stampato, se il tasto di console PRINT ALL è illuminato, ed il sistema visualizza il messaggio di errore ERROR 4.

Variabili numeriche

Una variabile numerica è un dato numerico, identificato con un nome, il cui valore è soggetto a cambiare durante l'esecuzione di una espressione. Nello stato calcoli immediati si possono utilizzare quattro variabili numeriche il cui nome è: Φ, ΦØ, Φ1, Φ2.

Per assegnare, da tastiera, un valore numerico ad una variabile, si deve premere il tasto [RESULT] ed, eventualmente, una delle tre cifre: Ø, 1 o 2; quindi il tasto [=] seguito dal numero da assegnare alla variabile.

Vediamo, ad esempio, come si assegna il valore 10 ad ognuna delle 4 variabili e cosa appare sul display come risultato di ogni assegnazione:

Quando si preme	Sul display appare
[RESULT] [=] (1) (0) [END OF LINE]	Φ =1Ø
[RESULT] (0) [=] (1) (0) [END OF LINE]	Φ Ø=1Ø
[RESULT] (1) [=] (1) (0) [END OF LINE]	Φ 1=1Ø
[RESULT] (2) [=] (1) (0) [END OF LINE]	Φ 2=1Ø

Per assegnare un valore ad una variabile come risultato dell'esecuzione di una espressione, si deve introdurre il nome della variabile seguita dal segno uguale e dall'espressione. Per esempio, premendo:

[RESULT] (2) [=] [RESULT] [+] [RESULT] (0) [+] [RESULT] (1) [−] [RESULT] (2) [END OF LINE]

il risultato della esecuzione dell'espressione Φ + Φ Ø + Φ 1 − Φ 2 è assegnato alla variabile Φ 2.

Vi sono due modi per visualizzare il valore corrente di una variabile. Con il primo si visualizza il valore della variabile e contemporaneamente si assegna tale valore alla variabile Φ ; con il secondo si visualizza soltanto il valore della variabile.

1. Per visualizzare il valore di una variabile ed assegnare il suo valore alla variabile Φ, si introduca il nome della variabile e si prema [END OF LINE]. Per esempio, per visualizzare il valore della variabile ΦØ ed assegnarlo alla variabile Φ si prema:
   [RESULT] (0) [END OF LINE]

2. Per visualizzare il valore di una variabile senza variare il valore della variabile Φ , si utilizzi il seguente formato: variabile=variabile e quindi si prema [END OF LINE]. Per esempio, per visualizzare il valore della variabile Φ senza modificare il valore di si prema: [RESULT] [0] [=] [RESULT] [0] [END OF LINE]

Nota: Ogni volta che si commuta il sistema nello stato calcoli immediati (premendo [CALC MODE]), le quattro variabili suddette sono eguagliate a zero. Se il sistema è nello stato calcoli immediati e si vogliono eguagliare a zero le quattro variabili contemporaneamente, si prema due volte [CALC MODE] --la prima volta si esce dallo stato calcoli immediati, la seconda volta vi si rientra.

La variabile Φ come totalizzatore

Se si introduce un valore numerico od una espressione numerica da tastiera premendo il tasto [SUM], invece del tasto [END OF LINE], la variabile Φ si comporta come un totalizzatore (accumulatore). Il valore introdotto od il risultato della espressione è aggiunto al valore precedente della variabile Φ.

L'impiego della variabile Φ come totalizzatore è mostrato negli esempi che seguono (si ricordi che quando si preme il tasto [CALC MODE], per passare nello stato calcoli immediati, tutte le variabili sono automaticamente eguagliate a zero).

Quando si preme	Si pone
[CALC MODE]	Φ=Ø, ΦØ=Ø, Φ1=Ø, Φ2=Ø
[RESULT] [=] [3] [END OF LINE]	Φ=3
[RESULT] [=] [9] [SUM]	Φ=12

Come si vede, premendo [SUM] si aggiunge 9 al precedente valore di Φ .

Quando si preme	Si pone
[CALC MODE]	Φ=Ø, ΦØ=Ø, Φ1=Ø, Φ2=Ø
[RESULT] [=] [1] [0] [END OF LINE]	Φ=1Ø ΦØ=Ø
[RESULT] [0] [=] [1] [0] [SUM]	Φ=2Ø ΦØ=1Ø

Come si vede, premendo [SUM] per assegnare il valore 1Ø a Φ Ø, si aggiunge anche 1Ø al precedente valore di Φ.

Quando si preme	Si pone			
[CALC MODE]	Φ=Ø,	ΦØ=Ø,	Φ1=Ø,	Φ2=Ø
[RESULT] [1] [=] [1] [0] [END OF LINE]		Φ=Ø	Φ1=1Ø	
[RESULT] [2] [=] [1] [0] [0] [END OF LINE]		Φ=Ø	Φ1=1Ø	Φ2=1ØØ
[RESULT] [=] [2] [0] [END OF LINE]		Φ=2Ø	Φ1=1Ø	Φ2=1ØØ
[RESULT] [=] [RESULT] [1] [+] [RESULT] [2] [SUM]		Φ=13Ø	Φ1=1Ø	Φ2=1ØØ

Come si vede, premendo [SUM] si aggiunge il valore di Φ1 + Φ2 (ossia 11Ø) al precedente valore di Φ (ossia 2Ø).

Quando si preme	Si pone			
[CALC MODE]	Φ=Ø,	ΦØ=Ø,	Φ1=Ø,	Φ2=Ø
[RESULT] [1] [=] [1] [0] [END OF LINE]		Φ=Ø	Φ1=1Ø	
[RESULT] [2] [=] [1] [0] [0] [END OF LINE]		Φ=Ø	Φ1=1Ø	Φ2=1ØØ
[RESULT] [=] [2] [0] [END OF LINE]		Φ=2Ø	Φ1=1Ø	Φ2=1ØØ
[RESULT] [2] [=] [RESULT] [1] [+] [RESULT] [2] [SUM]		Φ=13Ø	Φ1=1Ø	Φ2=11Ø

Come si vede, premendo [SUM] non solo si assegna il valore di Φ1 + Φ2 a Φ2, ma si aggiunge anche tale valore al precedente valore di Φ.

Per riassumere brevemente, il tasto [END OF LINE] serve per assegnare un valore ad una variabile; il tasto [SUM] serve sia per assegnare un valore ad una variabile che, contemporaneamente, per aggiungere tale valore al precedente valore della variabile Φ.

Operatori numerici

Gli operatori numerici definiscono quale operazione deve essere eseguita sui valori numerici degli operandi specificati. Essi producono come risultato un numero. Gli operatori numerici che si possono utilizzare sono:

Operatore numerico	Funzione
↑	Elevamento a potenza
/	Divisione
*	Moltiplicazione
+	Addizione e segno più
-	Sottrazione e segno meno

Le espressioni numeriche sono eseguite secondo il livello di priorità degli operatori che le costituiscono. Le operazioni con il più alto livello di priorità sono eseguite per prime; quelle con lo stesso livello di priorità sono eseguite nell'ordine da sinistra a destra.

Il P6066 osserva le regole dell'algebra per la definizione del livello di priorità di esecuzione di una operazione nell'ambito di una espressione numerica, per cui i livelli di priorità sono:

Operatore numerico	Livello di priorità
↑	Il più alto
* e /	↓
+ e -	Il più basso

Le parentesi (e) possono essere usate per cambiare l'ordine di esecuzione delle operazioni nell'ambito di una espressione numerica. Una espressione racchiusa tra parentesi è trattata come un singolo elemento numerico: viene valutata per ottenere il suo valore numerico, quindi tale valore è utilizzato per la valutazione della parte restante di una espressione più complessa di cui essa fa parte. Se più di una espressione è compresa tra parentesi, il calcolo inizia con la valutazione delle parentesi più interne. Nel seguito diamo ulteriori informazioni relative agli operandi numerici.

<u>Elevamento a potenza</u>

RESULT 1 ↑ RESULT 2	Indica che il valore di Φ1 deve essere elevato alla potenza di esponente dato da Φ2
Se Φ1 = Φ2=Ø	Φ1 ↑ Φ2 è uguale ad <u>uno</u>

Se Φ 1=Ø e Φ 2<Ø	Si ha una segnalazione di OVERFLOW
Se Φ 1<Ø e Φ 2 non è intero	Si ha una segnalazione di errore
Se Φ 1=Ø e Φ 2>Ø	Φ1 ↑ Φ 2 è uguale a <u>zero</u>

<u>Moltiplicazione ed addizione</u>

[RESULT] [1] [•] [RESULT] [2] equivale a [RESULT] [2] [•] [RESULT] [1]

[RESULT] [1] [+] [RESULT] [2] equivale a [RESULT] [2] [+] [RESULT] [1]

Come si vede, per la moltiplicazione e l'addizione, vale la proprietà commutativa.

[RESULT] [1] [•] [(] [RESULT] [2] [•] [RESULT] [)]

non equivale sempre a

[(] [RESULT] [1] [•] [RESULT] [2] [)] [•] [RESULT]

[RESULT] [1] [+] [(] [RESULT] [2] [+] [RESULT] [)]

non equivale sempre a

[(] [RESULT] [1] [+] [RESULT] [2] [)] [+] [RESULT]

perchè l'operazione tra parentesi in alcuni casi può dare un risultato troncato od arrotondato.

<u>Divisione e sottrazione</u>

[RESULT] [1] [/] [RESULT] [2] Indica Φ1 diviso per Φ2

Se Φ2=Ø Si ha segnalazione di OVERFLOW

[RESULT] [1] [-] [RESULT] [2] Indica Φ1 meno Φ2

<u>Segno</u>

[-] [RESULT] [+] [(] [-] [RESULT] [1] [)] [+] [RESULT] [2] [-] [(] [-] [RESULT] [0] [)]

è ammesso

[RESULT] [1] [+] [-] [RESULT] [2] non è ammesso

Il segno + ed il segno - possono essere usati dopo una parentesi aperta o prima di una espressione numerica.

<u>Nota</u>: Due operatori numerici non possono essere usati uno di seguito all'altro ma devono essere separati da parentesi.

Funzioni

Nello stato calcoli immediati si possono utilizzare funzioni numeriche di sistema o definite dall'utente.

Funzioni numeriche di sistema

Le funzioni numeriche di sistema possono essere utilizzate introducendo da tastiera il loro nome e (tra parentesi) l'argomento su cui si applicano. L'argomento può essere una costante numerica, una variabile numerica o, in generale, una espressione numerica.

Nella tabella seguente sono riassunte, in ordine alfabetico, le funzioni numeriche di sistema disponibili; la lettera X indica l'argomento della funzione.

Funzione numerica	Descrizione
ABS(X)	Valore assoluto di X
ACS(X)	Arcocoseno (in radianti) di X
ASN(X)	Arcoseno (in radianti) di X
ATN(X)	Arcotangente (in radianti) di X
COS(X)	Coseno di X radianti
COT(X)	Cotangente di X radianti
EXP(X)	Esponenziale in base e di X
HCS(X)	Coseno iperbolico di X radianti
HSN(X)	Seno iperbolico di X radianti
HTN(X)	Tangente iperbolica di X radianti
INT(X)	Il più grande numero intero minore od uguale ad X
LGT(X)	Logaritmo in base 10 di X
LOG(X)	Logaritmo naturale di X
RND	Numero casuale compreso tra zero ed uno
SGN(X)	Segno di X (+1 per X positivo, Ø per Ø, -1 per X negativo)

SIN(X)	Seno di X radianti
SQR(X)	Radice quadrata di X
TAN(X)	Tangente di X radianti

Tabella 6-1 Funzioni numeriche di sistema

Si possono inoltre utilizzare le seguenti funzioni:

LEN (string-exp) che ritorna il numero di caratteri di string-exp

SCN (string-exp, substring, n-occurrence, start-position) che ritorna la posizione della ennesima occorrenza in string-exp della sottostringa substring a partire dalla posizione specificata con start-position.

Funzioni definite dall'utente

Il P6066 permette di definire le funzioni di frequente impiego e di assegnarle ad uno degli 8 tasti della sezione funzioni definibili.

Si possono assegnare due funzioni per ogni tasto suddetto (utilizzando il tasto [SHIFT]) quindi si ha la possibilità di avere contemporaneamente fino a 16 funzioni. Ogni funzione può essere definita con al massimo 73 caratteri, ma il numero totale dei caratteri che definiscono tutte le funzioni assegnate ai tasti funzione non può essere maggiore di 238.

Vediamo con un esempio come si definisce ed assegna una funzione ad un tasto funzione, mentre il sistema è nello stato calcoli immediati.

Supponiamo di voler calcolare le radici reali di una equazione di secondo grado del tipo ax +bx-c utilizzando i tasti funzione. Si deve procedere nel seguente modo. Alla variabile Φ 2 si assegneranno i coefficienti di tipo a. Alla variabile Φ1 si assegneranno i coefficienti di tipo b. Alla variabile Φ Ø si assegneranno i coefficienti di tipo c. Quindi alla chiave F1 si assegni la stringa:
(-Φ1 + SQR(Φ1 * Φ1 - 4 * Φ2* ΦØ))/(2 * Φ2)
premendo i tasti [SHIFT] [FKEY#] [1] [.] e di seguito i tasti equivalenti alla stringa suddetta, seguiti da [END OF LINE]
Alla chiave F2 si assegni la stringa:

(- Φ1 - SQR(Φ1 * Φ1 - 4 * Φ2 * ΦØ))/(2 * Φ2)

premendo i tasti [SHIFT] [FKEY#] (2) (.) e di seguito i tasti equivalenti alla stringa suddetta, seguiti da [END OF LINE]

Per calcolare le radici reali di una equazione particolare, si introducano da tastiera i valori dei coefficienti assegnandoli nell'ordine alle variabili:

Φ2 = a
Φ1 = b
ΦØ = c

Dopo di che premendo: [RESULT] [=] [F1] [END OF LINE]

si ottiene, assegnandolo a Φ , il valore di una radice.

Premendo: [RESULT] [=] [F2] [END OF LINE]

si ottiene il valore della seconda radice.

Con analoghe procedure si possono assegnare ai tasti funzione altre espressioni numeriche. Premendo un solo tasto tali espressioni sono richiamate nel buffer di tastiera ed eventualmente inserite in espressioni più complesse.

Nota: Nello stato calcoli immediati si possono utilizzare, premendo i tasti funzione, le stesse espressioni assegnate ad essi dal programma presente in memoria principale.

Visualizzazione dei risultati

Il risultato della esecuzione di una espressione è visualizzato immediatamente sul display e, se il tasto di console [PRINT ALL] è acceso, stampato sul tabulato della stampante integrata. I risultati possono essere visualizzati (o stampati) in diversi formati che dipendono dalla posizione dell'indicatore dei decimali (vedi capitolo 1).

1. Quando l'indicatore dei decimali è nella posizione Ø (lo Ø è visibile nella relativa finestrella) non vengono visualizzate (stampate) le cifre decimali.

2. Quando l'indicatore dei decimali è in una posizione da 1 a 13 (la relativa cifra è visibile nella suddetta finestrella), vengono visualizzate (stampate)

da 1 a 13 cifre dopo il punto decimale se ciò è possibile.

3. Quando l'indicatore dei decimali è nella posizione Flt (Flt è visibile nella relativa finestrella), il numero è visualizzato nel formato interno ossia nella notazione scientifica (vedi paragrafo "Rappresentazione interna).

4. Quando l'indicatore dei decimali è nella posizione ST (ST è visibile nella finestrella):

   - i numeri interi sono visualizzati (stampati) con al massimo 8 cifre significative

   - i numeri con valore assoluto compreso tra 0.0999999995 e 99999999.4 sono visualizzati (stampati), con al massimo 8 cifre significative (se il numero è minore di 1 viene tralasciato lo zero che precede la parte decimale), nel formato in virgola fissa

   - i numeri con valore assoluto minore di 0.0999999995, che possono essere rappresentati con 8 cifre significative, sono visualizzati (stampati) nel formato in virgola fissa

   - tutti gli altri numeri sono rappresentati con al massimo 8 cifre significative nel formato in virgola mobile

Nota: L'indicatore dei decimali controlla anche il formato di visualizzazione e stampa del contenuto delle variabili numeriche Φ,ΦØ,Φ1,Φ2.

L'unità di misura degli angoli

Nel sistema P6066 gli angoli sono misurati normalmente in radianti; quindi, quando si forniscono gli argomenti alle funzioni trigonometriche, i valori introdotti sono assunti dal sistema come espressi in radianti. (Anche il valore ritornato dalla funzione ATN è espresso in radianti.)

Nello stato calcoli immediati è possibile scegliere l'unità di misura degli angoli, comunicando al sistema quella voluta, mediante la digitazione di uno dei seguenti comandi di predisposizione: SDEG, SGRAD, SRAD. (Quando si introduce SDEG e SGRAD, la luce di console

DEG GRAD si accende. Essa rimane accesa finchè si introduce SRAD, si esce dallo stato calcoli immediati, oppure il P6066 è spento.) Se si preme [S][D][E][G] , i valori assegnati come argomenti alle funzioni trigonometriche sono misurati in gradi sessagesimali. (Anche il valore ritornato dalla funzione ATN è espresso in gradi sessagesimali.)

Si noti che i valori assegnati agli argomenti delle funzioni trigonometriche, dopo aver introdotto SDEG, non possono essere espressi in gradi, primi e secondi di grado, ma in termini di gradi, decimi, centesimi, millesimi etc. di grado.

Se si preme [S][G][R][A][D] , i valori assegnati come argomenti alle funzioni trigonometriche sono interpretati in gradi centesimali. (Anche il valore ritornato dalla funzione ATN è espresso in gradi centesimali.)

Se si preme [S][R][A][D], i valori assegnati agli argomenti delle funzioni trigonometriche sono interpretati in radianti. (Anche ATN ritorna un valore espresso in radianti.) Se si preme due volte [CALC MODE] si ottiene lo stesso risultato, ma le variabili Φ, ΦØ, Φ1 e Φ2 sono eguagliate a zero.

Esempi di calcoli immediati

1. I seguenti esempi mostrano come l'impiego delle parentesi influisce sulla valutazione di una espressione:

Premendo	Sul display appare
[4][+][3][•][2][-][6][/][2][END OF LINE]	7
[(][4][+][3][)][•][2][-][6][/][2][END OF LINE]	11
[(][4][+][3][•][2][-][6][)][/][2][END OF LINE]	2
[(][(][4][+][3][)][•][2][-][6][)][/][2][END OF LINE]	4
[2][↑][3][↑][2][END OF LINE]	64

(2)(↑)(()(3)(↑)(2)()[END OF LINE] 512

2. L'esempio seguente mostra come si può calcolare la media di cinque numeri:

Premendo	
[RESULT][=](0)[END OF LINE]	Φ è uguagliata a zero
(5)[SUM]	Φ = 5
(6)[SUM]	Φ = 11
(1)(4)[SUM]	Φ = 25
(3)[SUM]	Φ = 28
(1)(2)[SUM]	Φ = 40
[RESULT][/](5)[END OF LINE]	Φ = 8

3. Negli esempi seguenti si mostra l'effetto prodotto posizionando l'indicatore dei decimali nelle posizioni indicate:

Premendo	Con l'indicatore dei decimali su	Sul display appare
[P][I][END OF LINE]	Ø	3
[P][I][END OF LINE]	1	3.1
[P][I][END OF LINE]	2	3.14
[P][I][END OF LINE]	13	3.1415926535900
[P][I][END OF LINE]	ST	3.1415927
[P][I][END OF LINE]	Flt	3.141592653590E+00

7. LO STATO DI DEBUGGING

Premessa

Fornendo una verifica sintattica immediata di ogni linea introdotta da tastiera e fornendo una verifica della coerenza sintattica tra le istruzioni di un intero programma, durante la fase di pre-esecuzione, il P6066 riduce il tempo necessario per individuare, diagnosticare e correggere gli errori in un programma. Quando, tuttavia, un programma è eseguito, è necessario determinare se funziona correttamente. Vi possono essere, infatti, degli errori di tipo logico nella sua struttura.

Per verificare la validità di un programma, lo si deve eseguire con dei dati di prova che producano dei risultati predefiniti. Si paragonano, quindi, questi ultimi con i risultati prodotti dal P6066. Se i risultati ottenuti non coincidono con quelli previsti ed il programma è breve o abbastanza semplice, si può, di solito, trovare la causa degli errori analizzando il programma stesso: l'ordine delle istruzioni, la correttezza del suo algoritmo, etc.; ma, se un programma è lungo e complesso, il semplice metodo di paragonare i risultati ottenuti con quelli previsti non sempre permette di trovare la causa degli errori: sarà necessaria una ulteriore analisi.

Lo stato di debugging del P6066 offre un insieme completo di strumenti per l'analisi e la verifica dei programmi. Nello stato di debugging, l'operatore ha un completo controllo sulla esecuzione di un programma. Si può fermare e riprendere l'esecuzione del programma, si possono visualizzare e modificare i valori delle variabili, si può controllare il flusso logico del programma, si può, insomma, vedere e tracciare l'esecuzione del programma.

Come accedere allo stato di debugging

Il sistema commuta nello stato di debugging quando: (1) si esegue il comando PREPARE, (2) viene eseguita una istr zione STOP, (3) si rileva un errore di tipo recuperabile (vedi appendice D) durante l'esecuzione di un programma,

oppure (4) si preme il tasto di console STEP.
Quando il sistema è nello stato di debugging, il tasto di console STEP è illuminato.

1. Il comando PREPARE (vedi capitolo 3) può essere introdotto durante lo stato comandi oppure durante lo stato calcoli immediati. Il programma presente in memoria principale è pre-eseguito ed il sistema commuta nello stato di debugging. Il tasto di console STEP si illumina.

2. Quando si esegue l'istruzione STOP, l'esecuzione del programma di cui essa fa parte è interrotta ed il sistema commuta nello stato di debugging. Sul display appare il messaggio STOP IN LINE line-num (dove line-num è il numero di linea dell'istruzione STOP).

3. Se, durante l'esecuzione di un programma, viene rilevato un errore di tipo recuperabile, l'esecuzione del programma è interrotta, il sistema commuta nello stato di debugging e viene visualizzato un appropriato messaggio di errore sul display.

4. Quando si preme il tasto di console STEP, l'esecuzione di un programma viene interrotta. Il sistema commuta nello stato di debugging e sul display appare il messaggio STEP IN LINE line-num (dove line-num è il numero di linea della successiva istruzione da eseguire). Se si preme di nuovo il tasto STEP, viene eseguita l'istruzione successiva, il sistema commuta di nuovo nello stato di debugging e un nuovo messaggio STEP IN LINE line-num appare sul display.

Strumenti dello stato di debugging

Quando il sistema è nello stato di debugging, si possono utilizzare i seguenti strumenti per ricercare la causa degli errori di un programma presente in memoria principale:

1. Comandi dello stato di debugging

   START
   STOP

2. Tasti di console

   CONTINUE
   STEP
   TRACE

3. Prestazioni dello stato calcoli immediati.

4. Analisi ed impiego delle variabili globali di programma.

5. Impiego delle funzioni definite nel programma.

Comandi dello stato di debugging

I comandi START e STOP possono essere utilizzati durante lo stato di debugging per verificare parti di un programma senza doverlo eseguire completamente.

1. Comando START
   Il comando START comunica al sistema di riprendere l'esecuzione di un programma, precedentemente interrotto, da un numero di linea specificato. La sintassi del comando è:

   STA [RT] line-num

   dove line-num è il numero di linea della istruzione da cui riprende l'esecuzione del programma.

2. Comando STOP
   Il comando STOP comunica al sistema di interrompere l'esecuzione del programma presente in memoria principale dopo l'esecuzione di una istruzione predefinita e di commutare nello stato di debugging. La sintassi del comando è:

   STO [P] line-num

   dove line-num è l'istruzione in cui si interrompe l'esecuzione del programma. Si noti che se si introducono successivamente due comandi STOP line-num, l'ultima introduzione invalida la predisposizione alla sospensione della esecuzione comandata dalla precedente. Se nel programma non esiste una istruzione con il numero di linea line-num specificato nel comando STOP, oppure durante l'esecuzione non è mai incontrata una istruzione con tale numero di linea, l'esecuzione prosegue fino al termine

senza che sia prodotta alcuna segnalazione. Si noti infine che il comando STOP line-num è un comando di predisposizione per cui quando è introdotto da tastiera l'esecuzione del programma non riprende; per far ripartire l'esecuzione del programma si deve premere CONTINUE o STEP o introdurre il comando START.

Tasti di console

Tre tasti di console sono particolarmente utili come strumenti di verifica dei programmi: CONTINUE , STEP e TRACE .

Nel seguito è spiegato il loro impiego e sono dati alcuni suggerimenti che permettono di utilizzare, nello stato di debugging, le prestazioni offerte dai tasti di console NO PRINT e BREAK .

Se mentre il sistema è nello stato di debugging, è premuto il tasto di console CONTINUE , riprende l'esecuzione del programma che è in memoria principale. Il tasto CONTINUE si illumina quanto è premuto mentre il sistema è nello stato di debugging. La funzione del tasto è attiva solamente quando il sistema si trova nello stato di debugging.

Se mentre il sistema è nello stato di debugging, è premuto il tasto di console STEP , si può eseguire passo a passo (ossia istruzione per istruzione) il programma presente in memoria principale. Ogni volta che una istruzione del programma è eseguita il sistema visualizza sul display il messaggio STEP IN LINE line-num, che specifica quale istruzione del programma sarà eseguita alla successiva pressione del tasto STEP . Dopo di che, ogni volta che si preme il tasto STEP , viene eseguita la successiva istruzione. (Si ricordi che il tasto di console STEP si illumina ogni volta che il sistema è nello stato di debugging.)

Il tasto di console TRACE permette di stampare il numero di linea di ogni istruzione eseguita, nell'ordine con cui essa è eseguita. Analizzando la stampa prodotta attivando la funzione TRACE si può seguire il flusso di esecuzione del programma. (Il tasto TRACE è acceso quando la funzione omonima è attiva.) Per utilizzare la funzione TRACE nello stato di debugging:

1. Premere il tasto TRACE .

2. Premere il tasto CONTINUE oppure introdurre il comando START.

Come conseguenza, si riprende l'esecuzione del programma e vengono stampati i numeri di linea delle istruzioni eseguite. Si noti che non vengono stampati i numeri di linea delle istruzioni non esecutive, come REM o DCL, e l'esecuzione del programma è realizzata come quando la funzione TRACE non è attiva -- per esempio, se una istruzione INPUT è eseguita, l'esecuzione del programma è interrotta e sul display appare un punto interrogativo che indica che il sistema è in attesa di dati da tastiera. (La funzione TRACE può essere attivata anche durante lo stato comandi, premendo TRACE e quindi introducendo il comando RUN.)

Quando la funzione del tasto NO PRINT è attiva, la stampante integrata è inibita; quindi vengono soppresse tutte le stampe richieste dal programma (mediante le istruzioni PRINT, PRINT USING, MAT PRINT e MAT PRINT USING) che è eseguito. Mentre si verifica un programma può essere utile sopprimere tali stampe -- così facendo si risparmia tempo di stampa e carta. Quando la funzione NO PRINT è attiva, il tasto NO PRINT è illuminato.

Il tasto BREAK, premuto nello stato di debugging, permette di terminare l'esecuzione di un programma e commuta il sistema nello stato comandi. Nello stato comandi, si può modificare il programma presente in memoria principale oppure introdurre un qualsiasi comando di sistema. Quando la funzione BREAK è attiva, il tasto BREAK è illuminato.

Prestazioni dello stato calcoli immediati

Quando il sistema è nello stato di debugging, si possono utilizzare tutte le prestazioni dello stato calcoli immediati -- si possono eseguire delle espressioni immediatamente, si possono usare le variabili Φ, ΦØ, Φ1 e Φ2 etc. (Si veda il capitolo 6 per una descrizione completa delle prestazioni dello stato calcoli immediati.) Si noti che, tuttavia, ogni operazione tipica dello stato calcoli immediati è eseguita nello stato di debugging senza premere il tasto CALC MODE che non è operativo nello stato di debugging.

Analisi ed impiego delle variabili globali di un programma

Durante lo stato di debugging si può analizzare il contenuto di tutte le variabili globali del programma presente in memoria principale. Per determinare il contenuto di una variabile globale, si digiti il suo nome da tastiera (ed eventualmente gli indici, se si tratta di una variabile con indice), quindi si prema il tasto [END OF LINE]. Come risultato, il valore della variabile è visualizzato e stampato (se la funzione PRINT ALL è attiva.)

Per esempio, per visualizzare il valore della variabile A1 si prema: [A] [1] [END OF LINE]

Per visualizzare il valore della variabile con indice B(2,5), si prema:

[B] [(] [2] [,] [5] [)] [END OF LINE]

Si possono assegnare, da tastiera, valori alla variabili globali di un programma, modificando i loro valori correnti, sia direttamente che come risultato dell'esecuzione di una espressione.

Vediamo due esempi:

1. Per assegnare il valore 5 alla variabile globale A, si prema:

   [A] [=] [5] [END OF LINE]

2. Per assegnare il risultato dell'espressione

   B*Φ1+SQR(C) alla variabile A, si prema:

   [A] [=] [B] [*] [RESULT] [1] [+] [S] [Q] [R] [(] [C] [)] [END OF LINE]

Nell'esempio A, B e C sono variabili globali del programma che è in memoria principale.

Nota: Nello stato di debugging non si può assegnare un valore ad una variabile locale. Per la definizione di variabile locale e di variabile globale si veda l'istruzione DEF/FNEND.

Impiego delle funzioni definite nel programma

Durante lo stato di debugging si possono utilizzare non solo le funzioni (numeriche o stringa) di sistema, ma anche le funzioni mono-linea o multi-linea definite nel programma presente in memoria principale.

Vediamo due esempi:

1. Per assegnare il valore della funzione mono-linea FNA, definita nel programma presente in memoria principale, alla variabile A, si prema:

   A = F N A (8 , 9) END OF LINE

2. Per assegnare il prodotto del valore della funzione mono-linea FNA per la funzione multilinea FNC, alla variabile B, si prema:

   B = F N A (3 , 5) * F N C (8 , 7) END OF LINE

Un esempio di debugging di un programma

In questo paragrafo diamo un esempio della tecnica impiegata per verificare, correggere ed ampliare un programma. Viene descritto l'impiego delle prestazioni del P6066 e l'interazione che si realizza tra l'utente, il sistema ed il linguaggio BASIC in una tipica sessione di debugging. I numeri sul margine sinistro della stampa ottenuta dal sistema corrispondono alle note che seguono nella descrizione. Le note spiegano cosa è stato fatto e perchè sono utilizzate le prestazioni dello stato di debugging descritte.

Supponiamo che un programmatore voglia realizzare un programma che genera tutti i numeri primi compresi in un certo campo predefinito. Come primo passo, per verificare l'algoritmo da utilizzare, introduce da tastiera un programma che calcola e stampa tutti i numeri primi esistenti tra 1 e 30. L'algoritmo utilizzato è il seguente: si assegnano ad una variabile I, in sequenza, i numeri interi compresi tra 1 e 30; ogni valore della variabile I è diviso, in sequenza, con i valori assunti da una variabile J (che variano da 2 alla radice quadrata di I); se il resto della divisione è zero, allora I non è numero primo, quindi viene assegnato ad I il numero I+1 e ripetuto il medesimo processo. Se il resto della divisione di I per i valori di J compresi tra 2 e la radice quadrata di I è diverso da zero, allora I è un numero primo e viene stampato; ad I è assegnato il numero I+1 e si riprende l'intero processo dall'inizio. Si impiegano i valori di J da 2 alla radice quadrata di I perchè se vi sono dei fattori primi di I questi saranno uno di tali valori.

```
NEW
10REM PROGRAMMA CHE CALCOLA I NUMERI PRIMI
20   PRINT" ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30 "
25PRINT
30 PRINT
40FORI=2TO30
50K=SQR(I)
60FORJ=2TOK
70E=I/J-INT(1/J)
80IFE=0THEN100
90NEXTJ
100PRINTI;
110NEXTI
120PRINT
125    PRINT
130PRINT" FINE DELL'ELENCO "
140END
(1) RUN
**** FORMALLY CORRECT PROGRAM ****
 ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30

(2) 2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23
 24  25  26  27  28  29  30

 FINE DELL'ELENCO
(3) LIST
FILE

0010 REM PROGRAMMA CHE CALCOLA I NUMERI PRIMI
0020 PRINT " ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30 "
0025 PRINT
0030 PRINT
0040 FOR I=2 TO 30 STEP 1
0050 LET K=SQR(I)
0060 FOR J=2 TO K STEP 1
0070 LET E=I/J-INT(1/J)
0080 IF E=0 THEN 100
0090 NEXT J
0100 PRINT I;
0110 NEXT I
0120 PRINT
0125 PRINT
0130 PRINT " FINE DELL'ELENCO "
0140 END

END OF LISTING

(4) FETCH 40
0040 FOR I=2 TO 30 STEP 1
0040 FOR I=1 TO 30 STEP 1
(5) RUN
**** FORMALLY CORRECT PROGRAM ****
 ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30

 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22
 23  24  25  26  27  28  29  30

 FINE DELL'ELENCO
(6) FETCH 70
0070 LET E=I/J-INT(1/J)
0070 LET E=I/J-INT(I/J)
RUN
**** FORMALLY CORRECT PROGRAM ****
 ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30

(7) 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22
 23  24  25  26  27  28  29  30

 FINE DELL'ELENCO
```

1. Il programmatore ha introdotto da tastiera il suo programma di prova e lo ha fatto eseguire dal sistema introducendo il comando RUN. Come indicato dal messaggio: **** FORMALLY CORRECT PROGRAM **** (il tasto PRINT ALL è attivo), il sistema non ha rilevato alcun errore durante l'analisi di coerenza sintattica tra le istruzioni del programma.

2. Uno sguardo alla stampa prodotta dal programma è sufficiente per vedere che i numeri generati non sono i numeri primi da 1 a 30, ma l'elenco completo dei numeri interi da 2 a 30.

3. Il programmatore introduce il comando LIST per ottenere un listing del suo programma più facile da leggere. Nota un primo errore nella istruzione con numero di linea 40; la variabile I assume come valore iniziale 2 invece di 1.

4. Introduce il comando FETCH per trasferire nel buffer di tastiera l'istruzione suddetta che viene corretta.

5. Riesegue il programma. Questa volta il numero 1 è compreso nell'elenco dei numeri stampati, ma sono ancora presenti anche i numeri non primi. Una successiva analisi del listing del programma gli permette di notare un errore nella istruzione 70: ha digitato un 1 invece di I.

6. Introduce di nuovo un comando FETCH per correggere l'istruzione suddetta.

7. Una successiva esecuzione del programma produce gli stessi errori.

8. A questo punto il programmatore decide di utilizzare gli strumenti dello stato di debugging, per cui introduce il comando PREPARE ed il sistema passa nello stato di debugging.

9. Preme il tasto di console STEP (come indicato dal messaggio STEP IN LINE). Il programmatore continua a premere il tasto STEP, facendo eseguire ogni volta una istruzione. Ad un certo punto, egli osserva che l'istruzione 100 -- che dovrebbe stampare solamente i numeri primi -- viene eseguita ogni volta.

10. Per verificare il valore corrente della variabile I (il valore stampato dall'istruzione 100), digita I da tastiera e preme [END OF LINE]. Sul display appare il numero 4 -- che non è un numero primo. Ora il programmatore può restringere la causa dell'errore a tre possibilità:

    - il calcolo nell'istruzione 70 è errato

    - il test nell'istruzione 80 è errato

    - il riferimento all'istruzione a cui saltare (contenuto nell'istruzione 80) è errato

    Rianalizzando il programma da questi punti di vista si avvede che è il terzo punto la causa di errore: se E=0 (cioè, se la divisione non fornisce alcun resto), l'esecuzione del programma dovrebbe proseguire dall'istruzione 110 per incrementare di 1 la variabile I. Il programmatore preme il tasto [BREAK] per commutare il sistema nello stato comandi e modificare il programma.

11. Introduce il comando FETCH per trasferire l'istruzione 80 nel buffer di tastiera. L'istruzione viene modificata.

12. Per assicurarsi che ora l'esecuzione del ciclo FOR/NEXT è corretta, il programmatore preme il tasto di console [TRACE] e quindi introduce il comando RUN.

13. Visto che la quarta volta in cui è eseguito il ciclo FOR/NEXT, l'istruzione 100 non è eseguita (è eseguita l'istruzione 110), preme di nuovo il tasto [TRACE] per inibirne la funzione. Il programma è così eseguito per intero, senza stampare i numeri di linea delle successive istruzioni eseguite.

14. Analizzando la stampa prodotta dal programma osserva che i numeri generati sono effettivamente tutti i numeri primi compresi tra 1 e 30.

15. Per avere una copia del programma corretto introduce di nuovo il comando LIST. Ora che i risultati prodotti dalla routine sono corretti, modifica il programma per ottenere una soluzione più generale del problema: genera un programma che produce tutti i numeri primi compresi tra due numeri interi N ed

M. Aggiunge le due istruzioni seguenti:

```
15 DISP " INTRODUCI I LIMITI N,M "
16 INPUT N,M
```

e modifica le istruzioni 20 e 40.

```
(8) PREPARE
    :: ROOM=28208 ::
    STEP        IN LINE 20
     ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30
(9) STEP        IN LINE 25

    STEP        IN LINE 30

    STEP        IN LINE 40
    STEP        IN LINE 50
    STEP        IN LINE 60
    STEP        IN LINE 100
    STEP        IN LINE 110
    STEP        IN LINE 50
    STEP        IN LINE 60
    STEP        IN LINE 100
    STEP        IN LINE 110
    STEP        IN LINE 50
    STEP        IN LINE 60
    STEP        IN LINE 100
    STEP        IN LINE 110
    STEP        IN LINE 50
    STEP        IN LINE 60
    STEP        IN LINE 70
    STEP        IN LINE 80
    STEP        IN LINE 100
    STEP        IN LINE 110
(10) I
     4
     1  2  3  4
(11) FETCH 80
     0080 IF E=0 THEN 100
     0080 IF E=0 THEN 110
(12) RUN
     **** FORMALLY CORRECT PROGRAM ****
      ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30

     #30

     #40
     #50
     #60
     #100
     #110
     #50
     #60
     #100
     #110
```

```
#50
#60
#100
#110
#50
#60
#70
#80
#110
#50
#60
#70
#80
#90
#100
#110
(13) #50
(14) 1  2  3  5  7  11  13  17  19  23  29

     FINE DELL'ELENCO
```

16. Il programmatore introduce il comando RESEQUENCE prima di richiedere un listing finale del programma.

17. Stampa il nuovo listing. Infine registra nella libreria MAT il programma con il nome NUMPR, eseguendo il comando SAVE NUMPR,MAT.

```
(15) LIST
     FILE

     0010 REM PROGRAMMA CHE CALCOLA I NUMERI PRIMI
     0020 PRINT " ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30 "
     0025 PRINT
     0030 PRINT
     0040 FOR I=1 TO 30 STEP 1
     0050 LET K=SQR(I)
     0060 FOR J=2 TO K STEP 1
     0070 LET E=I/J-INT(I/J)
     0080 IF E=0 THEN 110
     0090 NEXT J
     0100 PRINT I;
     0110 NEXT I
     0120 PRINT
     0125 PRINT
     0130 PRINT " FINE DELL'ELENCO "
     0140 END

     END OF LISTING


     15 DISP " INTRODUCI I LIMITI N,M ";
     16 INPUT N,M
     FETCH 20
     0020 PRINT " ECCO TUTTI I NUMERI PRIMI COMPRESI TRA 1 E 30 "
     0020 PRINT " ECCO TUTTI I NUMERI PRIMI COMPRESI TRA N ED M "
     FETCH 40
     0040 FOR I=1 TO 30 STEP 1
```

```
0040 FOR I=N TO M STEP 1
(16) RESEQUENCE
(17) LIS
FILE
0010 REM PROGRAMMA CHE CALCOLA I NUMERI PRIMI
0020 DISP " INTRODUCI I LIMITI N,M ";
0030 INPUT N,M
0040 PRINT " ECCO TUTTI I NUMERI PRIMI COMPRESI TRA N ED M "
0050 PRINT
0060 PRINT
0070 FOR I=N TO M STEP 1
0080 LET K=SQR(I)
0090 FOR J=2 TO K STEP 1
0100 LET E=I/J-INT(I/J)
0110 IF E=0 THEN 140
0120 NEXT J
0130 PRINT I;
0140 NEXT I
0150 PRINT
0160 PRINT
0170 PRINT " FINE DELL'ELENCO "
0180 END

END OF LISTING


SAVE   NUMPR, MAT
```

8. IMPIEGO DEI TASTI FUNZIONE

Agli otto tasti funzione della sezione funzioni definibili, vedi figura 8-1, si possono assegnare, utilizzando anche il tasto [SHIFT], fino a 16 funzioni definite dall'utente.

Questo capitolo riprende in modo organico i diversi aspetti sull'impiego dei tasti funzione, già descritti nei capitoli precedenti (vedi capitolo 3 comando LDKEYS, capitolo 5 istruzione FKEY# e capitolo 6), per permettere al lettore di ritrovare facilmente le informazioni che gli permettono di sfruttare in modo completo questa prestazione importante del sistema P6066.

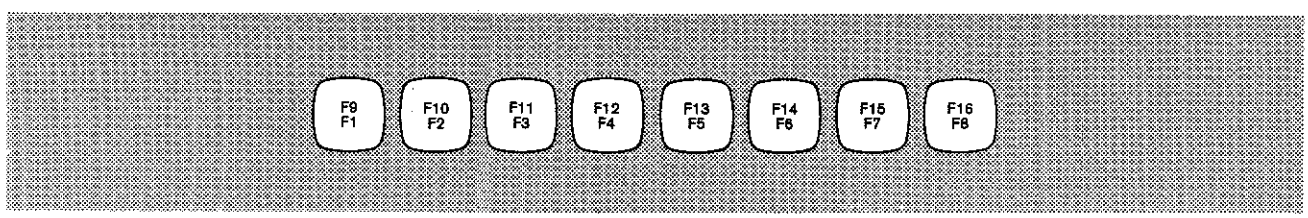


Figura 8-1 Tasti funzione

Assegnazione di una funzione ad un tasto funzione

L'assegnazione di una funzione ad un tasto funzione può essere effettuata durante tre diversi stati del sistema, ma la struttura dell'assegnazione è sempre la seguente:

FKEY# n, stringa

dove

n
è un numero intero, da 1 a 16, che indica a quale tasto funzione (F1 ÷ F16) deve essere assegnata la stringa specificata

stringa
indica quale espressione numerica o stringa di caratteri deve essere assegnata al tasto funzione specificato.

L'impiego della assegnazione di una funzione ad un tasto funzione varia in relazione allo stato del sistema. Gli stati del sistema in cui si può assegnare ai tasti funzione una funzione specifica sono:

- lo stato d'esecuzione programma
- lo stato calcoli immediati e lo stato di debugging
- lo stato comandi — mediante il comando LDKEYS

Ad ogni tasto funzione si può assegnare una stringa di caratteri, od una espressione numerica, con non più di 73 caratteri. Le stringhe di caratteri, o le espressioni numeriche, assegnate ai tasti funzione, sono registrati in un registro (buffer) di 238 caratteri (byte); quindi il numero totale di caratteri assegnati a tutti i tasti funzione non può essere maggiore di 238. Si noti che viene usato sempre lo stesso registro per registrare le stringhe di caratteri, o espressioni numeriche, assegnate ai tasti funzione, qualunque sia lo stato del sistema in cui tale assegnazione è effettuata. Per cui, le stringhe di caratteri, o espressioni numeriche, assegnate ai tasti funzione durante lo stato calcoli immediati, saranno cancellate quando verrà eseguito un programma in cui sono presenti delle istruzioni di assegnazione di funzione ai tasti funzione.

Nota: Per ricordare l'assegnazione fatta ai tasti funzione, si può utilizzare una striscia di carta con 16 caselle nelle quali si possono scrivere dei codici mnemonici relativi ai tasti funzione sottostanti (vedi figura 1-8).

Assegnazione di una funzione ai tasti funzione durante l'esecuzione di un programma

I tasti funzione sono utilizzati, in genere, in programmi costituiti da diverse routine affinchè l'operatore possa scegliere la routine da far eseguire. (Naturalmente si può utilizzare un programma per assegnare ai tasti funzione delle funzioni che saranno poi utilizzate nello stato calcoli immediati o nello stato di debugging.) Per assegnare da programma una stringa di caratteri o una espressione numerica ai tasti funzione si deve utilizzare l'istruzione FKEY#(vedi capitolo 5).

Quando l'istruzione FKEY# è eseguita, la stringa di caratteri a destra del segno di virgola è assegnata al tasto funzione specificato (quindi è memorizzata nel relativo registro). Dopo di che, quando il tasto funzione è premuto la stringa di caratteri ad esso assegnata è trasferita nel buffer di tastiera.

Diamo di seguito due esempi di impiego dei tasti funzione a cui è assegnata una funzione predefinita da programma.

Esempio 1: Vediamo come si crea un programma che permette di usare i tasti funzione in modo da poter scegliere quale routine nell'ambito del programma deve essere eseguita. Non diamo il programma completo ma solo la parte che evidenzia l'impiego delle istruzioni FKEY#. L'impiego dei tasti funzione permetterà di digitare valori di lunghezza, area e volume espressi con qualsiasi unità di misura, poichè opportune routine li convertiranno nelle unità di misura accettate dall'algoritmo "principale" del programma.

```
0010 FKEY #1, START 2000:
0020 FKEY #2, START 2110:
0030 FKEY #3, START 2210:
0040 FKEY #4, START 2310:
0050 FKEY #5, START 2410:
0060 FKEY #6, START 2510:
0070 FKEY #7, START 100:
0080 DISP "Scegli la routine ";
0090 STOP
0100 REM DATI IN METRI, METRI QUADRI, METRI CUBI
0110 INPUT L,A,V
0120
...  ...   ...  }
...  ...   ...  }        ALGORITMO PRINCIPALE
...  ...   ...  }

2000 REM DATI IN CENTIMETRI, CENTIMETRI QUADRI, CENTIMETRI CUBI
2010 INPUT L,A,V

...  ...   ...  }
...  ...   ...  }        ROUTINE DI CONVERSIONE
...  ...   ...  }

2100 GOTO 120
2110 REM DATI IN DECIMETRI, DECIMETRI QUADRI, DECIMETRI CUBI
2120 INPUT L,A,V

...  ...   ...  }
...  ...   ...  }        ROUTINE DI CONVERSIONE
...  ...   ...  }
```

```
2200 GOTO 120
2210 REM DATI IN MILLIMETRI, MILLIMETRI QUADRI, MILLIMETRI CUBI
2220 INPUT L,A,V

...  ...   ...  }
...  ...   ...  }        ROUTINE DI CONVERSIONE
...  ...   ...  }

2300 GOTO 120
2310 REM DATI IN MILLIMETRI, CENTIMETRI QUADRI, CENTIMETRI CUBI
2320 INPUT L,A,V

...  ...   ...  }
...  ...   ...  }        ROUTINE DI CONVERSIONE
...  ...   ...  }

2400 GOTO 120
2410 REM DATI IN MILLIMETRI, CENTIMETRI QUADRI, MILLIMETRI CUBI
2420 INPUT L,A,V

...  ...   ...  }
...  ...   ...  }        ROUTINE DI CONVERSIONE
...  ...   ...  }

2500 GOTO 120
2510 REM DATI IN CENTIMETRI,MILLIMETRI QUADRI, MILLIMETRI CUBI
2520 INPUT L,A,V

...  ...   ...  }
...  ...   ...  }        ROUTINE DI CONVERSIONE
...  ...   ...  }

2600 GOTO 120
2610 END
```

Le prime istruzioni del programma sono delle assegnazioni ai tasti funzione del comando START che permettono di riprendere l'esecuzione del programma da istruzioni prefissate. Dopo di che segue una istruzione STOP che interrompe l'esecuzione del programma. L'utente potrà così, premendo il tasto funzione adeguato (da F1 a F7), lanciare l'esecuzione della routine di input che traduce i valori di lunghezza, area e volume (introdotti con una certa unità di misura) in metri, metri quadri e metri cubi. Dopo la routine di conversione è eseguito l'algoritmo principale del programma.

Nota: I due punti ":" alla fine di ogni assegnazione (istruzione 10 ÷ 70) sono interpretati dal sistema come un END OF LINE per cui il comando START è introdotto nel sistema premendo solo il tasto funzione relativo senza dover premere successivamente [END OF LINE] .

Esempio 2: Se si vuole usare il sistema per convertire le misure inglesi nel sistema decimale e viceversa, si può scrivere un programma come il seguente:

```
0010 FKEY#1,*0.3048:
0020 FKEY#9,/0.3048:
0030 FKEY#2,*25.4:
0040 FKEY#10,/25.4:
0050 FKEY#3,*1.6099344:
0060 FKEY#11,/1.609344:
0070 FKEY#4,*0.092903:
0080 FKEY#12,/0.092903
0090 FKEY#5,*6.4516:
0100 FKEY#13,/6.4516:
0110 FKEY#6,*28.3168:
0120 FKEY#14,/28.3168:
0130 FKEY#7,*4.54609:
0140 FKEY#15,/4.54609:
0150 FKEY#8,*0.45360:
0160 FKEY#16,/0.45360:
0170 END
```

Quando il programma è eseguito le espressioni di conversione sono assegnate ai tasti funzione. Si può quindi utilizzare il sistema nello stato esecuzione di calcoli immediati per convertire il valore introdotto da tastiera premendo successivamente il rispettivo tasto funzione. Si avrà che:

Premendo	il tasto	si convertono	in
"	F 1	piedi	metri
"	F 2	pollici	mm
"	F 3	miglia	Km
"	F 4	piedi quadri	m^2
"	F 5	pollici quadri	cm^2
"	F 6	piedi cubi	litri
"	F 7	galloni	litri
"	F 8	libbre	Kg
"	F 9	metri	piedi
"	F10	millimetri	pollici
"	F11	chilometri	miglia
"	F12	metri quadri	piedi quadri
"	F13	centimetri quadri	pollici quadri
"	F14	litri	piedi cubi
"	F15	litri	galloni
"	F16	chilogrammi	libbre

Assegnazione di una funzione ai tasti funzione durante lo stato calcoli immediati

Quando il sistema è nello stato calcoli immediati si possono assegnare ai tasti funzione delle espressioni o costanti utilizzate comunemente. L'assegnazione è effettuata come già descritto nel capitolo 6.

Esempio: Si usi il sistema P6066 per calcolare il valore medio e la deviazione standard di una serie di numeri introdotti da tastiera. Si fanno le seguenti assegnazioni ai tasti funzione:

```
FKEY#1,$=
FKEY#2,$1=$1+$:
FKEY#3,$2=$2+$↑2:
FKEY#4,$0=$0+1:
FKEY#5,$1/$0:
FKEY#6,SQR[($0*$2-$1↑2)/$0/($0-1)]:
FKEY#10,$1=0:
FKEY#11,$2=0:
FKEY#12,$0=0:
```

dove F 2 calcola la sommatoria dei valori X introdotti da tastiera

" F 3 calcola la sommatoria di X^2

" F 4 registra quanti numeri (Y) compongono la media

" F 5 calcola la media

" F 6 calcola la deviazione standard dalla media

" F1Ø azzera Σ X

" F11 azzera Σ X^2

" F12 azzera Y (il numero di dati che compongono la media)

La procedura è la seguente:

1 - Premere F10
2 - Premere F11 } si azzerano gli accumulatori
3 - Premere F12
4 - Premere F1
5 - Introdurre X
6 - Premere [END OF LINE] si assegna X a ΦØ

7 - Premere F2
8 - Premere F3
9 - Premere F4
1Ø - Si ripeta la procedura dal punto 4 per tutti i valori di X
11 - Per avere la media si preme F5
12 - Per avere la deviazione standard si preme F6
13 - Per ripetere i calcoli su un altro insieme di valori X si riparte dal numero 1.

Impiego dei tasti funzione nello stato comandi

Durante lo stato comandi si possono registrare sul disco sistema o sul floppy disk sistema le stringhe di caratteri od espressioni numeriche che sono state assegnate ai tasti funzione. In questo modo si possono ripristinare ai tasti funzione le funzioni desiderate. Se l'assegnazione di una funzione ad un tasto funzione è effettuata da programma, non è necessaria la suddetta operazione perchè ogni volta che il programma è eseguito ai tasti funzione vengono riassegnate le funzioni definite nel programma. Se però l'assegnazione di un contenuto ai tasti funzione è effettuata durante lo stato calcoli immediati, per poterla ripristinare, si ricorre al seguente comando di sistema:

STKEYS

che registra su disco sistema o sul floppy disk sistema il contenuto attuale dei tasti funzione. Quando il sistema è successivamente inizializzato, il buffer relativo ai tasti funzione viene caricato dal disco sistema o dal floppy disk sistema con il contenuto ivi registrato mediante il comando STKEYS. Se tale contenuto associato ai tasti funzione è modificato, sia da un programma che da nuove assegnazioni ai tasti funzione effettuate durante lo stato calcoli immediati, esso può essere ripristinato mediante il comando LDKEYS (vedi capitolo 3). I comandi suddetti sono utili quando si ha una notevole mole di gestione delle librerie. In questo caso si assegnano ai tasti funzione i comandi che richiamano in memoria principale ed eseguono i programmi di utilità che gestiscono le librerie. Quindi si registrano le assegnazioni suddette sul disco sistema o sul floppy disk sistema mediante il comando STKEYS. Quando le assegnazioni registrate sul disco sistema sono necessarie, si possono richiamare in memoria principale mediante il comando LDKEYS. Dopo di che non è necessario digitare ogni volta i comandi, richiesti, perchè basta premere l'appropriato tasto

funzione per richiamare in memoria principale il programma di utilità.

9. IMPIEGO DI UN VIDEO DISPLAY

Il P6066 può essere collegato ad un video display Olivetti DSM 6680 (vedi figura 9-1) utilizzando due diversi governi: DSC 6681 e DSC 6683. Il governo DSC 6681 permette solamente la visualizzazione di linee di caratteri alfanumerici, mentre il governo DSC 6683 permette inoltre di tracciare dei grafici e delle immagini comunque complesse.

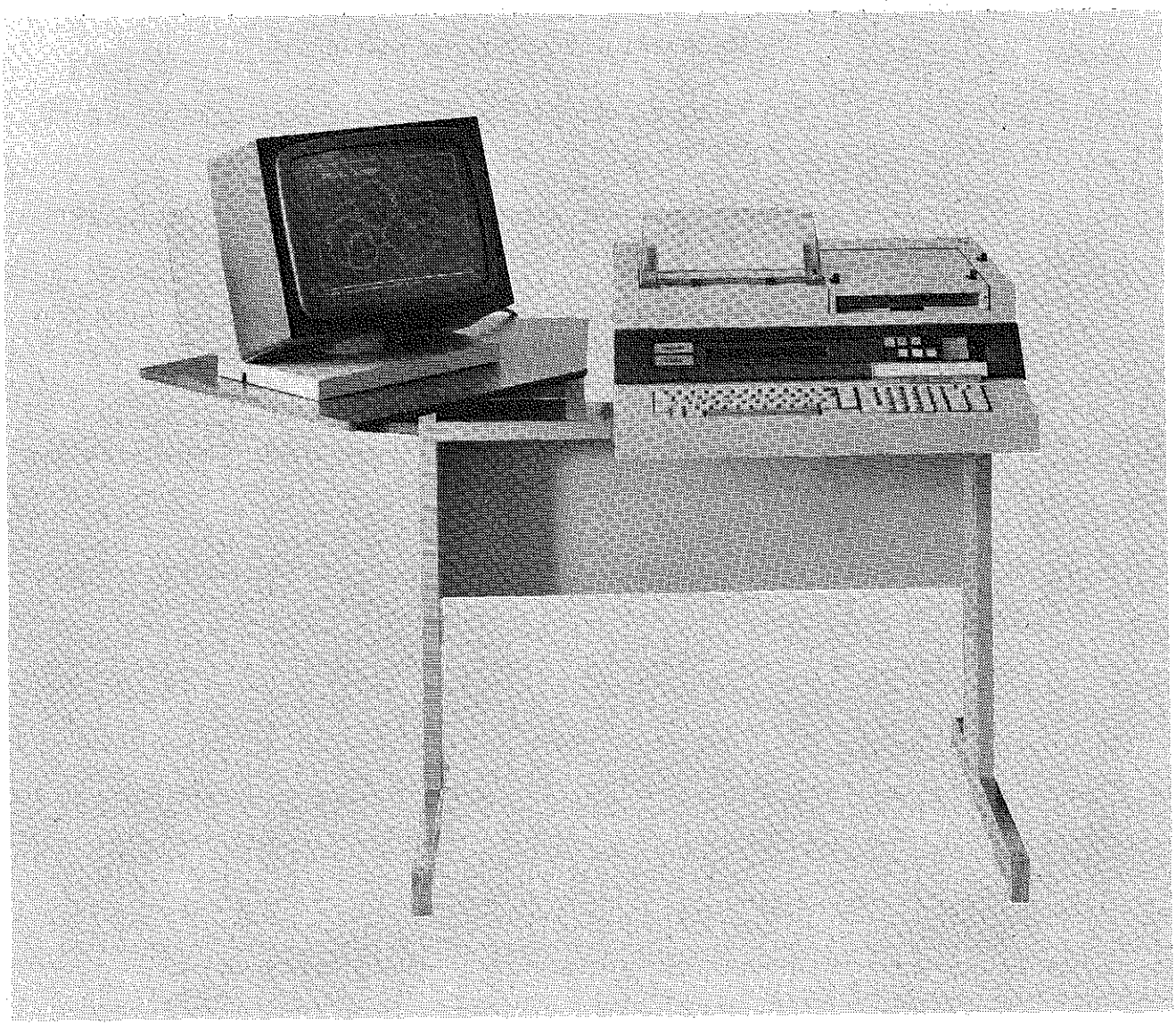


Figura 9-1 P6066 con video display Olivetti DSM 6680

In questo capitolo è descritto l'impiego del video display suddetto con il governo DSC 6681, per quanto riguarda il tracciamento di grafici si veda il manuale P6066 Prestazioni grafiche - Manuale del programmatore GP Code 3978350 Q (0).

Attivazione del video display

Per utilizzare il video display deve essere stato eseguito un comando CONFIGURE con l'operando EXD.

Informazioni visualizzate sullo schermo

Sullo schermo si possono visualizzare fino a 41 linee contemporaneamente, con un massimo di 80 caratteri per linea. Sullo schermo possono apparire:

- messaggi di sistema e messaggi di errore
- caratteri introdotti da tastiera
- testi e dati specificati in istruzioni DISP e DISP USING
- testi e dati specificati in istruzioni PRINT, PRINT USING, MAT PRINT e MAT PRINT USING
- informazioni fornite dai comandi CATALOG, ENVIRONMENT, LIST, LVTOC e dal programma di utilità FLPRINT

Per facilitare il riferimento al contenuto dello schermo supponiamo che le linee siano numerate da 0 a 40, come nella figura 9-2.

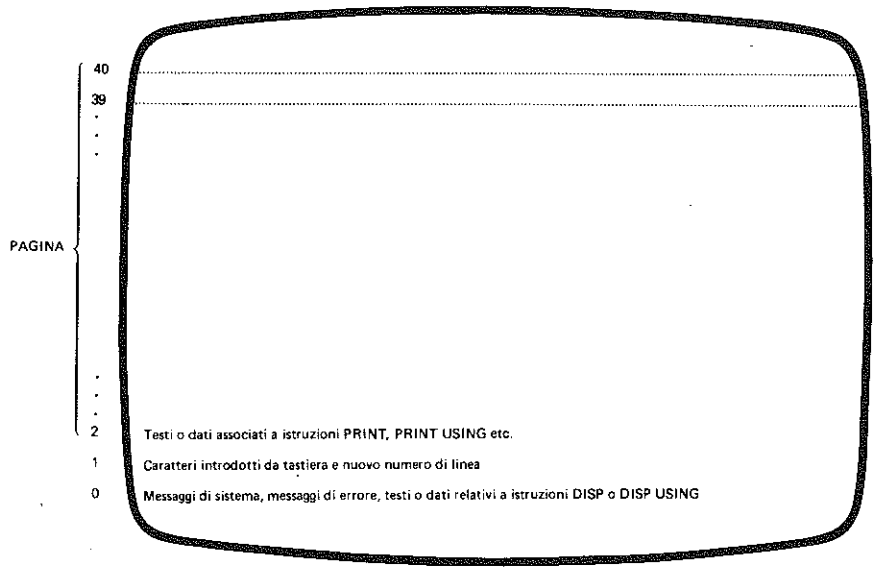


Figura 9-2 Informazioni visualizzate sul video display

Sulla linea 0 sono visualizzati tutti i messaggi di sistema, i messaggi di errore ed i testi o dati specificati nelle istruzioni DISP e DISP USING. Se PRINT ALL è attivo, il contenuto della linea 0 è contemporaneamente visualizzato sulla linea 2 ed i precedenti contenuti delle linee da 2 a 39 sono spostati di una linea verso l'alto.

Sulla linea 1 sono visualizzati i caratteri introdotti da tastiera durante la generazione o l'editing di un programma o file testo, oppure in risposta ad un'istruzione di INPUT. Nella linea è visualizzato anche un cursore che indica la posizione in cui sarà inserito il successivo carattere digitato. Se la linea digitata non è corretta, quando è premuto END OF LINE essa rimane nella posizione precedente per essere corretta. Nel caso in cui sia corretta, scompare dallo schermo dopo che si è premuto END OF LINE , od è trasferita sulla linea 2 se PRINT ALL è attivo ed i precedenti contenuti delle linee da 2 a 39 sono spostati di una linea verso l'alto.

Si noti infine che, se è stata attivata la numerazione automatica, ogni volta che una linea corretta è accettata dal sistema il numero di linea successivo è vi-

sualizzato all'inizio della linea 1.

Sulla linea 2 sono visualizzati i testi ed i dati specificati nelle istruzioni PRINT, PRINT USING, MAT PRINT e MAT PRINT USING. In questo caso i contenuti precedenti delle linee da 2 a 39 sono spostati verso l'alto di una linea.

Sulle linee da 2 a 40 sono visualizzate le informazioni fornite dai comandi CATALOG, ENVIRONMENT, LIST, LVOTC e dal programma di utilità FLPRINT. Inoltre possono essere visualizzate stringhe di caratteri specificate in un'istruzione DISP con un formato particolare che descriveremo più avanti.

Impaginazione

Le linee da 2 a 40 compongono una "pagina" di 39 linee che possono essere visualizzate contemporaneamente. Se le informazioni dirette allo schermo, da un programma od un comando, sono costituite da più di 39 linee, alcune linee saranno cancellate e non si potranno mai leggere. Per evitare questo inconveniente si deve utilizzare il tasto NO PRINT. Infatti, attivando la funzione NO PRINT, tasto NO PRINT illuminato, oltre ad ottenere un risparmio di carta sulla stampante integrata, in quanto se ne inibisce il funzionamento, s'interrompe l'esecuzione del programma o del comando di sistema dopo che sullo schermo sono apparse le 39 linee della prima pagina. Il tasto CONTINUE s'illumina per avvertire l'utente di questa situazione e permettergli di leggere le informazioni visualizzate. Premendo il tasto CONTINUE l'esecuzione del programma o del comando riprende dal punto interrotto e continua fino alla fine o finchè una nuova pagina di 39 linee non sia visualizzata sullo schermo. Quando questo avviene l'esecuzione s'interrompe nuovamente ed il tasto CONTINUE s'illumina di nuovo. L'utente può così leggere tutte le pagine che sono visualizzate fino alla fine, premendo successivamente il tasto CONTINUE.

Istruzioni BASIC e comandi per il video display

Per l'impiego del video display, oltre alle istruzioni BASIC ed ai comandi menzionati precedentemente, si possono utilizzare una nuova forma di istruzione DISP ed i comandi ERASE e REVERSE che sono descritti nelle pagine seguenti.

Istruzione DISP

Funzione

Permette la visualizzazione di una stringa di caratteri in una linea qualsiasi del video display.

Formato

DISP "control-char op-code [line-position char-position] [char-string]"

dove:

control-char
è il carattere θ

op-code
è uno dei seguenti caratteri: ~~K,J,~~ B,A,S

line-position
è un carattere della tabella ISO usato per rappresentare la posizione di una linea sul video

char-position
è un carattere della tabella ISO usato per rappresentare la posizione di un carattere in una linea sul video

char-string
è una stringa di caratteri che deve essere visualizzata sul video

Azione

Quando l'istruzione è eseguita con ~~K~~ come op-code, ~~la linea~~ nella posizione line-position è completamente sostituita con ~~una linea~~ contenente la stringa char-string nella posizione specificata con char-position. ~~La stringa è preceduta e seguita da eventuali~~ spazi.

Quando l'istruzione è eseguita con ~~J~~ come op-code, l'istruzione funziona come nel caso precedente, ma la linea è visualizzata invertendo il suo colore con

quello dello sfondo dello schermo e viceversa.

Quando l'istruzione è eseguita con B come op-code, sono soppresse le visualizzazioni del contenuto delle istruzioni PRINT che seguono nel programma. Se è specificato l'operando char-string la stringa corrispondente è visualizzata nella linea 0 (vedi la figura 9-2).

Quando l'istruzione è eseguita con A come op-code, sono riattivate le visualizzazioni del contenuto delle istruzioni PRINT che seguono nel programma. Se è specificato l'operando char-string la stringa corrispondente è visualizzata nella linea 0 (vedi figura 9-2).

Quando l'istruzione è eseguita con S come op-code, sposta di una linea verso l'alto le linee da 2 a 39 con perdita della linea 40 (vedi figura 9-2). Se è specificato l'operando char-string la stringa corrispondente è visualizzata nella linea 0.

Note

1. Per introdurre il carattere di controllo θ si devono premere contemporaneamente i tasti [CONTROL] e [I].

2. La posizione della linea è specificata mediante un carattere della tabella ISO il cui valore decimale corrispondente varia da 32 a 70 (vedi l'Appendice E). La linea dello schermo in cui sarà visualizzata la stringa di caratteri è quella il cui numero è uguale al valore decimale corrispondente al carattere specificato meno 30. Quindi le stringhe di caratteri possono essere visualizzate su una linea qualunque da 2 a 40 come mostrato in figura 9-2.

3. La posizione da cui è visualizzata una stringa di caratteri è specificata mediante un carattere della tabella ISO il cui valore decimale corrispondente varia da 32 a 111 (vedi l'Appendice E). La posizione nella linea da cui inizierà la stringa di caratteri è la posizione data dal valore decimale corrispondente al carattere specificato meno 31. Quindi in ogni linea si possono visualizzare fino a 80 caratteri.

4. La posizione di una linea e la posizione d'inizio di una stringa di caratteri nell'ambito di una linea possono essere specificate usando la funzione

di sistema CHR$. Se si utilizza CHR$ l'istruzione DISP avrà il seguente formato:

```
DISP "control-char op-code" =CHR$(30+n) +CHR$(31+m)  +  "char-string"
```

dove n è un intero da 2 a 40 ed m è un intero da 1 80. Il valore specificato per n corrisponde ad una linea del video (vedi figura 9-2); il valore specificato per m è la posizione nella linea in cui inizierà la stringa di caratteri. Per ulteriori informazioni sulla funzione CHR$ si veda il capitolo 4.

5. Se è specificata una stringa di caratteri che supera la 80esima posizione di una linea, la stringa è troncata alla 80esima posizione.

6. La stringa di caratteri può essere specificata come variabile stringa (vedi l'esempio 3).

Esempi

1. La seguente istruzione DISP visualizza la stringa di caratteri:

   Calcolo matriciale

   nella linea 10 iniziando dalla posizione 9

   100 DISP "θ⁄⁄((Calcolo matriciale"

2. In questo caso viene usata la funzione di sistema CHR$ per visualizzare la stringa di caratteri:

   Calcolo matriciale

   nella linea 10 iniziando dalla posizione 9

   100 DISP "θ⁄⁄"+CHR$(30+10)+CHR$(31+9)+"Calcolo matriciale"

3. Nel seguente esempio il valore di A$ (che è la stringa di caratteri Calcolo matriciale) è visualizzato nella linea 10 iniziando dalla posizione 9.

   100 DISP "θ⁄⁄((" +A$

Comando ERASE

Funzione	Cancella completamente il contenuto dello schermo del video.
Formato	**ERA[SE]**
Azione	Il comando ERASE cancella completamente il contenuto presente sullo schermo del video.

Comando REVERSE

Funzione	Converte il colore dei caratteri visualizzati nel colore dello sfondo dello schermo e viceversa.
Formato	**REV[ERSE]**
Azione	REVERSE converte il colore dei caratteri visualizzati nel colore dello sfondo dello schermo e viceversa.